EPSON

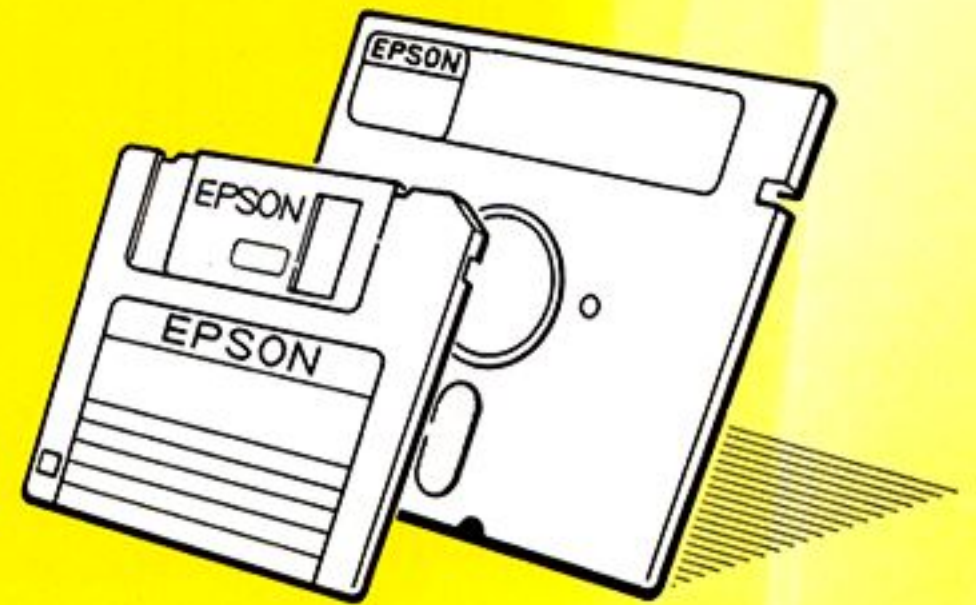
日本語DiskBASIC

reference manual

リファレンスマニュアル●

コマンド・ステートメント・関数
エラーメッセージ

付録

PC286ML10
Y161991137 01

ご 注 意

(1) 本書の内容の一部、または全部を無断で転載することは、固くお断りします。

(2) 本書の内容について、将来予告なしに変更することがあります。

(3) 本書の内容については万全を期して作成致しましたが、万一誤り・お気付きの点がありましたら、ご連絡くださいますようお願いいたします。

(4) 運用した結果の影響については、(3)項にかかわらず責任を負いかねますのでご了承ください。

本製品を日本国外へ持ち出す場合のご注意

本製品は「外国為替及び外国貿易管理法」に定める戦略物資(または役務)に該当します。したがって、本製品を輸出する場合には同法に基づく日本国政府の輸出許可が必要です

EPSON

日本語DiskBASIC

reference manual

リファレンスマニュアル●

はじめに

日本語 Disk BASIC

日本語 Disk BASIC は、基本的な BASIC をもとに強力な日本語処理機能、グラフィック機能、スクリーンエディタ機能、倍精度計算機能などを拡張した BASIC です。日本語 Disk BASIC を理解することによって、自分で自由にプログラムを作成して、さまざまな処理を行わせることができます。

日本語 Disk BASIC のマニュアルは次の 2 冊から構成されています。

日本語 Disk BASIC ユーザーズマニュアル
日本語 Disk BASIC リファレンスマニュアル

日本語 Disk BASIC ユーザーズマニュアル

日本語 Disk BASIC ユーザーズマニュアルは日本語 Disk BASIC を機能ごとに解説したもので、次の 3 部構成になっています。

解説編

日本語 Disk BASIC でプログラムを作成するのに必要な事柄を機能別に解説しています。具体的には日本語 Disk BASICの起動と終了の方法、プログラム作成のための基礎知識、キーボードやグラフィック機能などを解りやすく説明しています。

BASIC ユーティリティ編

日本語 Disk BASIC ユーティリティディスクに入っているユーティリティソフト「BMENU」と「SYSSET」の使用方法について説明しています。BMENU によってディスクのフォーマットやメモリスイッチの変更などを行うことができます。

付録

日本語 Disk BASIC の補足的な技術資料をまとめて説明しています。PC-286U/US シリーズのサウンド機能についても解説しています。

日本語 Disk BASIC リファレンスマニュアル

日本語 Disk BASIC リファレンスマニュアルは、日本語 Disk BASIC を構成する命令を具体的に解説したものです。BASIC の命令にはコマンド・ステートメント・関数の区別があり、これらを、引きやすさ、探しやすさを重視して、アルファベット順に説明しています。また、機能面からの検索の際に役立つように、機能別の索引もつけました。それぞれのケースに応じて活用してください。

これらのマニュアルは EPSON PC シリーズすべての共通マニュアルになっており、機種によっては使用できない機能も一緒に解説していますのでご注意ください。このような機能には次のものがあります。

サウンド機能	PC-286U/US シリーズで使用可能です。
数値演算プロセッサ	PC-286L/LE シリーズ以外の機種は、オプションで数値演算プロセッサを取り付けることができます。

目次

はじめに …… (1)

リファレンス編 …… 1

第1章　コマンド・ステートメント・関数 …… 3

第1章の見方 …… 3
ABS …… 5
AKCNV$ …… 6
ASC …… 7
ATN …… 8
ATTR$ …… 9
AUTO …… 10
BEEP …… 11
BLOAD …… 12
BSAVE …… 13
CALL …… 14
CDBL …… 15
CHAIN …… 16
CHR$ …… 18
CINT …… 19
CIRCLE …… 20
CLEAR …… 22
CLOSE …… 23
CLS …… 24
COLOR[1] …… 25
COLOR[2] …… 27
COLOR@ …… 29
COM ON/OFF/STOP …… 30
COMMON …… 31
CONSOLE …… 32
CONT …… 33
COPY …… 34
COS …… 35
CSNG …… 36
CSRLIN …… 37
CVI/CVD/CVS …… 38
DATA …… 39
DATE$ …… 40
DEF FN …… 41
DEFINT/DEFSNG/DEFDBL/DEFSTR …… 42
DEF SEG …… 43
DEF USR …… 44
DELETE …… 45
DIM …… 46
DRAW …… 47
DSKF …… 50
DSKI$ …… 51
DSKO$ …… 53
EDIT …… 54
END …… 55
EOF …… 56
ERASE …… 57
ERL/ERR …… 58
ERROR …… 59
EXP …… 60
FIELD …… 61
FILES/LFILES …… 62
FIX …… 63
FOR～TO～NEXT …… 64
FPOS …… 66
FRE …… 67
GET# …… 68
GET@ …… 69
GOSUB～RETURN …… 71
GOTO …… 72
HELP ON/OFF/STOP …… 73
HEX$ …… 74
IF～THEN～ELSE/IF～GOTO～ELSE …… 75

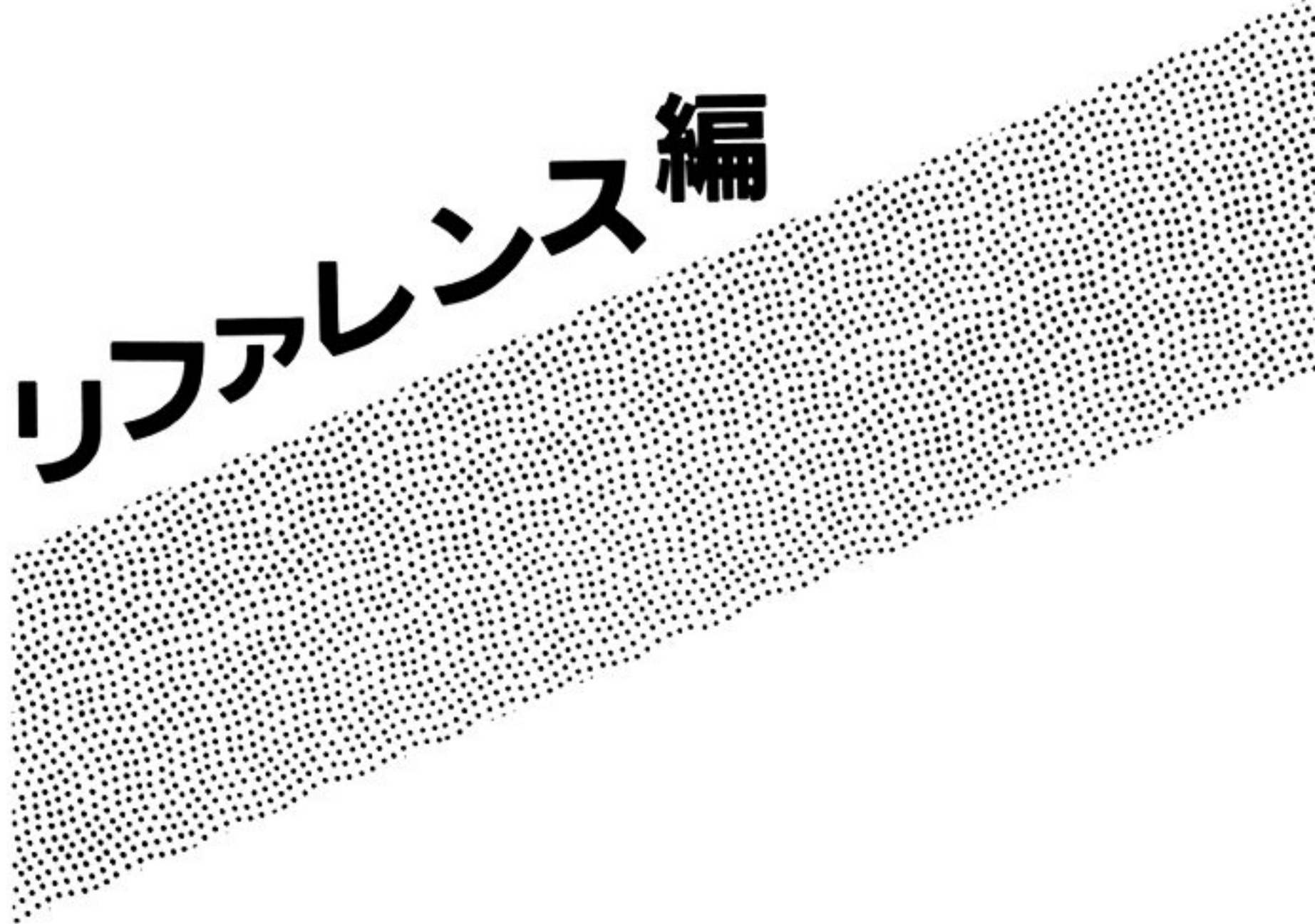
INKEY$ …… 76
INP …… 77
INPUT …… 78
INPUT# …… 79
INPUT$ …… 80
INPUT WAIT …… 81
INSTR …… 82
INT …… 83
JIS$ …… 84
KACNV$ …… 85
KEXT$ …… 86
KEY …… 87
KEY LIST …… 88
KEY ON/OFF/STOP …… 89
KILL …… 90
KINPUT …… 91
KINSTR …… 92
KLEN …… 93
KMID$ …… 94
KNJ$ …… 95
KPLOAD …… 96
KTYPE …… 97
LEFT$ …… 98
LEN …… 99
LET …… 100
LINE …… 101
LINE INPUT …… 103
LINE INPUT# …… 104
LINE INPUT WAIT …… 105
LIST/LLIST …… 106
LOAD …… 107
LOC …… 108
LOCATE …… 109
LOF …… 110
LOG …… 111
LPOS …… 112
LSET/RSET …… 113
MAP …… 114
MERGE …… 115
MID$ …… 116
MID$ …… 117
MKI$/MKS$/MKD$ …… 118
MON …… 119
NAME …… 120
NEW …… 121
NEW ON …… 122
OCT$ …… 123
ON COM GOSUB …… 124
ON ERROR GOTO …… 125
ON～GOSUB/ON～GOTO …… 126
ON HELP GOSUB …… 127
ON KEY GOSUB …… 128
ON PEN GOSUB …… 129
ON STOP GOSUB …… 132
ON TIME$ GOSUB …… 133
OPEN …… 134
OPTION BASE …… 136
OUT …… 137
PAINT …… 138
PEEK …… 141
PEN …… 142
PEN ON/OFF/STOP …… 143
POINT …… 144
POINT[1] …… 145
POINT[2] …… 146
POKE …… 147
POS …… 148
PRINT/LPRINT …… 149
PRINT USING/LPRINT USING …… 151
PRINT#/PRINT# USING …… 154
PSET/PRESET …… 156
PUT# …… 157
PUT@ …… 158
RANDOMIZE …… 160
READ …… 161
REM …… 162

RENUM …… 163
RESTORE …… 164
RESUME …… 165
RETURN …… 166
RIGHT$ …… 167
RND …… 168
ROLL …… 169
RUN …… 170
SAVE …… 171
SCREEN …… 172
SEARCH …… 177
SET …… 178
SGN …… 179
SIN …… 180
SPACE$ …… 181
SPC …… 182
SQR …… 183
STOP …… 184
STOP ON/OFF/STOP …… 185
STR$ …… 186
STRING$ …… 187
SWAP …… 188
TAB …… 189
TAN …… 190
TIME$ …… 191
TIME$ ON/OFF/STOP …… 192
TRON/TROFF …… 193
USR …… 194
VAL …… 195
VARPTR …… 196
VIEW …… 198
VIEW …… 200
WAIT …… 201
WHILE～WEND …… 202
WIDTH …… 203
WIDTH LPRINT …… 204
WINDOW …… 205
WINDOW …… 206
WRITE …… 207
WRITE# …… 208

第2章　エラーメッセージ …… 209

付録 …… 215

A　文字コード表 …… 217
B　予約語 …… 218
C　誘導関数 …… 220
D　機能別索引 …… 221

リファレンス編

第1章

コマンド・ステートメント・関数

第1章では、日本語Disk BASICを構成するコマンド・ステートメント・関数などを個々に説明しています。これらは、引きやすさ、探しやすさを重視して、アルファベット順になっていますが、機能面からの索引に役立つように、機能別の索引を付けています。それぞれのケースに応じて活用してください。

第1章の見方

コマンド・ステートメント・関数は次ページのような書式に従って説明していきます。

算術演算子、関係演算子、論理演算子については「日本語Disk BASICユーザーズマニュアル」を参照してください。

①名称　命令の名称を示します。

②属性　命令がコマンド・ステートメント・関数のいずれに属するのかを示します。

③読み方　略称を含め、いく通りかの読み方を示します。

④語源　省略されて用いられることが多いので、完全なつづりと元の英語の意味を実際の命令との関連で紹介します。

A

② 関数

① ASC

③ 【アスキー】

④ ASCII
米国情報交換用標準文字コード(ASCII)の短縮形。

⑤ 機　能　文字列の先頭の1文字の文字コードを与えます。

⑥ 書　式　ASC(文字列)

⑦ 使用例　A=ASC("ASCII")　←「A」の文字コード65をAに代入します。

⑧ 解　説
・文字列の先頭1文字(1バイト)の、文字コードを10進数で与えます。
・文字列の長さが0、すなわち空文字列(ヌルストリング)の場合は、Illegal function call(違法関数呼び出し)エラーになります。
・文字コードから、文字を得るにはCHR$関数を用います。
・2バイト文字の文字コード(JIS漢字コード)を調べるにはJIS$関数を用います。

⑨ 参　照　CHR$(文字コードを文字に変換)
JIS$(2バイト文字の文字コードを得る)

⑩ プログラム例

```
100 ' ASC
110 ' --- 文字を文字コードに変換 ---
120 A$="COMPUTER"
130 FOR I=1 TO LEN(A$)
140   B$=MID$(A$,I,1)
150   PRINT USING "! : ###";B$;ASC(B$)
160 NEXT
170 END

RUN
C :  67
O :  79
M :  77
P :  80
U :  85
T :  84
E :  69
R :  82
OK
```

ASC　7

⑤機能　命令の基本的な機能・使用する目的などを説明します。

⑥書式　命令の正しい構文を示します。その際用いている記号は、次のような意味を持ちます。

- 英語の大文字や#、$、カンマ'，'、丸かっこ'（'、'）'はそのままの形式で入力します。英語の大文字は小文字で入力しても構いません。
- それ以外の日本語などには任意のパラメータを指定します。パラメータの意味は解説を参照してください。
- 角かっこ［　］で囲まれた項目は任意に指定できる項目です。したがって場合によっては省略することもできます。
- 縦線｜　｜で囲まれた項目は、それぞれの中から1つを選択して指定します。
- ・・・に続く項目は、直前の項目を一行の範囲内で、必要な回数だけ繰り返して指定することができます。

⑦使用例　プログラムで使用する場合の見本を示します。

⑧解説　命令のそれぞれについて詳細な解説を行っています。

⑨参照　関連性の高いほかの命令や、関連する項目を紹介します。

⑩プログラム例　実行を前提としたプログラム例を示します。
プログラム例や実行例はプリンタで印字したものです。したがって画面上の表示と異なる場合がありますので注意してください。

ABS

【アブソリュート】

ABSolute
絶対的な、純粋という意味から絶対値を意味する。

機　能　絶対値を計算します。

書　式　ABS(数式)

使用例　A=ABS(−70)　　←−70の絶対値70を求めます。

解　説
- (　)内の**数式**の絶対値を計算します。
- ABS関数の結果の型と**数式**の型は一致します。ただし数式が整数でその値が−32768のときに限り、単精度実数の32768を返します。

プログラム例

```
100 ' ABS / SQR
110 ' ---  数値の絶対値をとり平方根を求める  ---
120 INPUT "数値は ";I
130 IF I<0 THEN I=ABS(I)
140 PRINT TAB(14);"絶対値 =";I,"平方根 =";SQR(I)
150 GOTO 120

RUN
数値は ? -12
              絶対値 = 12    平方根 = 3.4641
数値は ? 2
              絶対値 = 2     平方根 = 1.41421
数値は ?
```

AKCNV$

【エーケーコンバート】

Ank Kanji CoNVert
ANK文字(1バイト文字)を漢字(2バイト文字)に変換する意味。

機能　文字列に含まれる1バイト文字を、2バイト文字に変換します。

書式　AKCNV$(文字列)

使用例　A$=AKCNV$(B$)　　←文字列B$に含まれる1バイト文字を2バイト文字に変換し、A$に代入します。

解説
- ( )内の**文字列**中の1バイト文字を、2バイト文字に変換します。
- 変換した2バイト文字の前後にはKI(漢字シフトイン)、KO(漢字シフトアウト)コードを挿入します。

KIコード	&H1B4B
KOコード	&H1B48

参照　KACNV$(2バイト文字を1バイト文字に変換)

プログラム例

```
100 ' AKCNV$
110 ' ---   1バイト文字を2バイト文字へ変換   ---
120 K$="ＡＢＣABCアイウｱｲｳ漢字"
130 PRINT K$;"   --->   ";AKCNV$(K$)
140 END

RUN
ＡＢＣABCアイウｱｲｳ漢字   --->   ＡＢＣＡＢＣアイウアイウ漢字
OK
```

ASC

【アスキー】

ASCii
米国情報交換用標準文字コード(ASCII)の短縮形。

機　能　文字列の先頭の1文字の文字コードを与えます。

書　式　ASC(文字列)

使用例　A=ASC("ASCII")　←「A」の文字コード65をAに代入します。

解　説
- **文字列**の先頭1文字(1バイト)の、文字コードを10進数で与えます。
- **文字列**の長さが0、すなわち空文字列(ヌルストリング)の場合は、Illegal function call(違法関数呼び出し)エラーになります。
- 文字コードから、文字を得るにはCHR$関数を用います。
- 2バイト文字の文字コード(JIS漢字コード)を調べるにはJIS$関数を用います。

参　照　CHR$(文字コードを文字に変換)
JIS$(2バイト文字の文字コードを得る)

プログラム例

```
100 ' ASC
110 ' --- 文字を文字コードに変換  ---
120 A$="COMPUTER"
130 FOR I=1 TO LEN(A$)
140   B$=MID$(A$,I,1)
150   PRINT USING "! = ###";B$;ASC(B$)
160 NEXT
170 END

RUN
C =  67
O =  79
M =  77
P =  80
U =  85
T =  84
E =  69
R =  82
OK
```

ATN

【アークタンジェント】

Arc TaNgent
三角関数の $\tan^{-1}$（逆正接）の意味。

機能　逆正接（アークタンジェント）を計算します。

書式　ATN（数式）

使用例
PI＃＝3.14159265358979323846
A＝ATN(1.5)＊180/PI＃
←1.5の逆正接を計算し、それを度に変換します。

解説
- （　）内の**数式**の逆正接を計算します。
- ATN関数の結果はラジアンで、$-\pi/2$ から $\pi/2$ の範囲の値になります。ラジアンを度に変換するには$180/\pi$を掛けます。πには次のような値を使用すると最も精度の高い計算を行うことができます。
  倍精度実数（PI＃をπの値を持つ変数とします）
  PI＃＝3.14159265358979323846
- **数式**が倍精度実数のときは、結果も倍精度で計算した値になります。他の場合は、単精度で計算した値になります。

参照
COS（余弦）
SIN（正弦）
TAN（正接）

プログラム例

```
100 ' ATN
110 ' ---   ATNの値をラジアンと度で計算する   ---
120 PI=3.141592653589793#
130 FOR I=0 TO 3 STEP .5
140     PRINT I,ATN(I),180/PI*ATN(I)
150 NEXT I
160 END

RUN
 0              0              0
 .5             .463648        26.5651
 1              .785398        45
 1.5            .982794        56.3099
 2              1.10715        63.4349
 2.5            1.19029        68.1986
 3              1.24905        71.565
OK
```

ATTR$

【アトリビュート・ダラー】

ATTRibute
属性、特性の意味からドライブ、ファイルの属性を調べること。

〔機　能〕 指定したドライブやファイルの属性を調べます。

〔書　式〕 ATTR$(|ドライブ番号 / #ファイル番号 / ファイル指定子|)

〔使用例〕 PRINT ATTR$(1)　　←ドライブ1の属性文字を画面に表示します。

〔解　説〕
- **ドライブ番号**で指定したディスクドライブ中のディスク、**ファイル番号**で指定したオープンしているファイル、**ファイル指定子**で指定したディスク上のファイルの属性を調べます。
- 属性は3文字の文字列 **"REP"** で表し、それぞれ次の意味を示します。

属性文字	機　能	意　味
P	書き込み禁止	SET 文でP指定(書き込み禁止)をすると、ここにPを表示します。 指定しない場合は空白("　")になります。
E	プロテクトセーブ	SAVE コマンドでP指定(プロテクトセーブ)をすると、ここにEを表示します。 指定しない場合は空白("　")になります。
R	リード・アフタ・ライト	SET 文でR指定(リード・アフタ・ライト)をすると、ここにRを表示します。指定しない場合は空白("　")になります。

リード・アフタ・ライト　書き込みの際に、書き込んだ内容と、メモリ上の内容の比較チェックを行うこと。

〔参　照〕 SAVE(プログラムのセーブ)
SET(書き込み属性の設定・解除)

〔プログラム例〕

```
SAVE "TEST",P
OK
PRINT ATTR$("TEST")
 E
OK
```

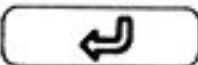
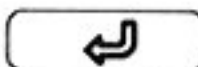
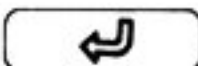
AUTO

【オート】

AUTOmatic
自動的、機械的の意味から、自動的に行番号を発生すること。

機 能 プログラムを作成する際に、自動的に行番号を発生します。

書 式 AUTO [最初の行番号] [, [増加分]]

使用例

AUTO ←最初の行番号を10とし、以後20, 30・・・と10間隔で行番号を表示していきます。

AUTO 100, 20 ←最初の行番号を100とし、以後120, 140・・・と20間隔で行番号を表示していきます。

解 説

- AUTOコマンドを実行すると、[↵]キーを押すたびに指定した内容で自動的に行番号を表示します。
- 行番号の発生は[STOP]キーまたは[CTRL]+[C]キーを押すと終了します。このときに表示していた行番号の行は、プログラムに含みません。終了後はBASICのコマンド入力待ちになります。
- **最初の行番号**や、**増加分**の指定を省略した場合は、それぞれ次のような行番号を発生します。

書 式	最初の行番号	増加分
省略	10	10
最初の行番号, 増加分	指定した行番号	指定した増加分
最初の行番号	指定した行番号	10
最初の行番号,	指定した行番号	直前のAUTOの増加分
, 増加分	10	指定した増加分

- **最初の行番号**にピリオド(.)を指定すると直前に修正、入力、表示した行を指定したことになります。
- 行番号の後ろにアスタリスク(*)を表示したときは、その行番号を持つ行がすでにプログラム中に存在することを示します。これに続けて何らかのデータを入力すると、その内容で行の内容を変更します。何も入力せずに[↵]キーを押すと、その行を削除します。
- AUTO文でプログラムを作成している間も、スクリーンエディタ機能を使ってプログラムの修正を行うことができます。修正後に[↵]キーを押すと、引き続き新しい行番号を表示します。

BEEP

【ビープ】

BEEP
警笛を鳴らすことから、本体のスピーカを鳴らすこと。

機　能　スピーカを鳴らします。

書　式　BEEP　[機能]

使用例　IF A>100　THEN BEEP　　←Aが100を超えると「ピー」とスピーカが鳴ります。

解　説
- **機能**が1でスピーカを鳴らし(鳴り続ける)、0で止めます。
- **機能**を省略したときは、PRINT CHR$(7)と同じで、一定時間スピーカを鳴らします。

機能	意味
1	スピーカを鳴らす
0	スピーカを止める
省略	PRINT CHR$(7)と同機能

プログラム例

```
100 ' BEEP
110 ' ---  BEEPを鳴らす  ---
120 INPUT "リターンキーを押す ";R$
130 BEEP
140 GOTO 120
```

BLOAD

【ビーロード】

Binary LOAD
binary は２進という意味で０と１のこと。すなわち、コンピュータが直接理解する機械語のこと。load は積む、乗せるという意味から、メモリに書き込むこと。

機能　機械語プログラムやデータなどのファイルを直接メモリに読み込みます(ロード)。

書式　BLOAD　"ファイル指定子" [, ロードアドレス] [, R]

使用例

```
10 DEF SEG=&H8000
20 BLOAD "mouse. cod"
```

←ファイルを読み込むセグメントアドレスを設定します。

←mouse. cod というファイルを読み込みます。

解説

- **ファイル指定子**で指定したファイルを**ロードアドレス**で指定したアドレスから順次読み込みます。
- **ロードアドレス**には－32768から32767(&H0000 から&HFFFF)までの数値または数式を指定します。このアドレスは、直前に実行した DEF SEG 文で指定したセグメントアドレスのオフセットアドレスです。
- **ロードアドレス**を省略すると、このファイルを BSAVE 文でセーブした際に指定した開始アドレスから読み込みます。
- R を指定するとプログラムをロードした後、ただちにプログラムの先頭から実行を開始します。このとき、すでにオープンしているファイルはクローズしません。
- BASIC は指定したアドレスから無条件にデータをロードしていきます。したがって、ロードアドレスに以下の領域を指定するとプログラムが暴走する可能性がありますので注意してください。

  BASIC のシステム領域
  BASIC のスタック領域
  BASIC の変数領域
  BASIC のプログラム領域
  メモリの範囲外など

参照

BSAVE(機械語ファイルのセーブ)
CALL(機械語プログラムの実行)
DEF SEG(セグメントアドレスの設定)
DEF USR(機械語プログラム実行開始アドレスの定義)
USR(機械語プログラムの実行)
日本語 Disk BASIC ユーザーズマニュアル　第10章　機械語プログラム

BSAVE

【ビーセーブ】

Binary SAVE

binaryは2進という意味。メモリの内容をそのままセーブすること。

機　能　メモリの内容を、そのままの形式でファイルに保存(セーブ)します。

書　式　BSAVE　"ファイル指定子", 開始アドレス, バイト数

使用例　BSAVE "DEMO. COD", 0, 255　←DEMO. COD というファイル名でアドレス0から255バイト分をセーブします。

解　説
- **ファイル指定子**で指定したファイルに、**開始アドレス**から、**バイト数**で指定したバイト数分をセーブします。
- **開始アドレス**には−32768から32767(&H0000 から&HFFFF)までの数値または数式を指定します。このアドレスは、直前に実行した DEF SEG 文で指定したセグメントアドレスのオフセットアドレスです。
- セーブする**バイト数**には−32768から32767(&H0000 から&HFFFF)までの数値または数式を指定します。このバイト数分のデータをセーブします。
- BASIC は指定したアドレスから無条件にデータをセーブしていきます。したがって、メモリ外のアドレスを指定しないよう注意してください。

参　照　BLOAD(機械語ファイルのロード)

DEF SEG(セグメントアドレスの設定)

日本語 Disk BASIC ユーザーズマニュアル　第10章　機械語プログラム

CALL

【コール】

CALL
呼ぶ。そのことから機械語プログラムを呼ぶ、つまり実行すること。

機　能　BASIC プログラムから機械語プログラムを実行します。

書　式　CALL　変数名[(引数 [, 引数・・・])]

使用例

```
100 CLEAR ,&H8000                            ←機械語プログラム領域の確保
110 DEF SEG=&H8000                           ←機械語プログラムセグメントの設定
120 BLOAD "mouse. cod"                       ←機械語プログラムの読み込み
130 MOUSE=0                                  ←機械語プログラムの開始アドレスを設定し
 ・                                             ます。
 ・
 ・
140 CALL MOUSE(AX%, BX%, CX%, DX%, ES%)      ←機械語プログラムを実行します。
```

解　説

- **変数名**に設定したアドレスから機械語プログラムを実行します。したがって、CALL 文を実行する前に、実行開始アドレスをこの**変数**に与えておきます。このアドレスはDEF SEG 文で設定したセグメントアドレスのオフセットアドレスです。
- **引数**は、機械語プログラムに渡す変数の名前です。これはカンマ(,)で区切って必要な数だけ指定します。**引数**の機械語プログラムへの引き渡し方法は、「第10章　機械語プログラム」を参照してください。
- BASIC プログラムから機械語プログラムを実行する方法として、ほかに USR 関数があります。

参　照

BLOAD(機械語プログラムのロード)
DEF SEG(セグメントアドレスの設定)
DEF USR(機械語プログラムの実行開始アドレスの定義)
USR(機械語プログラムの実行)
日本語 Disk BASIC ユーザーズマニュアル　第10章　機械語プログラム

CDBL

【コンバート・ダブル】

Convert DouBLe
２倍の、２重のという意味の double から倍精度、すなわち倍精度に変換(convert)すること。

機　能　整数、単精度実数を倍精度実数に変換します。

書　式　CDBL(数式)

使用例　A#＝CDBL(A!)^2　　←A! の２乗を倍精度で求めます。

解　説

- (　)内の**数式**の値を倍精度実数に変換します。
- CDBL 関数は、数値を倍精度実数型に型変換するだけです。値そのものは変化しません。
- 倍精度型変数に代入するときや、演算の一方が倍精度実数のときには自動的に倍精度実数への型変換を行います。したがって A#＝CDBL(3042.12!) と A#＝3042.12! の結果は同じになります。
- また単精度実数を変換すると、変換した倍精度実数と一致しないことがあります。これは単精度数のほうが、倍精度数よりも有効桁数が小さく、より大きな誤差を含んでいるからです。

例
```
PRINT CDBL(0.1!)
 .1000000014901161
OK
```

0.1#≒CDBL(0.1!)であることに注意してください。

参　照　CINT(整数に変換)
CSNG(単精度実数に変換)

CHAIN

【チェイン】

CHAIN
連鎖することから、プログラムをディスクから読み込み、実行すること。

機能 指定したプログラムファイルを読み込み、実行します。

書式 CHAIN [MERGE] ファイル指定子 [, 行番号] [, ALL] [, DELETE 削除範囲]

使用例

CHAIN "PROG"	←ドライブ1からPROGを読み込み、実行します。
CHAIN MERGE "2：PROG", 1000, ALL, 500-900	←メモリ上のプログラムとドライブ2のPROGを接続し1000行から実行します。

解説

- **ファイル指定子**で指定したファイルを読み込み、実行します。
- **行番号**はメモリ上のプログラムと、ディスク上のプログラムを読み込んだ後、実行を開始する行番号です。**行番号**を省略すると、読み込んだ後で一番小さい行番号から実行します。
- **MERGE**を指定をすると、メモリ中のプログラムとファイル指定子で指定したプログラムを結合して実行します。この場合、読み込むプログラムはアスキーセーブしたプログラムでなければなりません。メモリ中のプログラムで宣言した変数の型は、結合した後でも有効です。
- **MERGE**を省略すると、メモリ中のプログラムはファイル指定子で指定したプログラムで置き換わります。このとき、変数の型は読み込んだ後のプログラムでは無効になります。
- **ALL**を指定すると、メモリ中のプログラムで使用したすべての変数の内容を、読み込んだプログラムに引き渡します。
- **ALL**を省略すると、メモリ中のプログラムで、COMMON文で宣言した変数だけを引き渡します。
- **削除範囲**は**MERGE**を指定したときだけ意味を持ちます。連鎖する前に、メモリ中のプログラムの、**削除範囲**で指定した範囲を削除します。**削除範囲**の指定方法はDELETE文と同じように、行番号と行番号の間をハイフン(－)でつなげて表します。また行番号の代わりにラベルを使用することができます。
- ファイルは、CHAIN文の実行による影響を受けません。メモリ中のプログラムでオープンしたファイルはそのまま、読み込んだプログラムに引き継ぎます。
- 読み込んだプログラムに、ON ERRORなどのONで始まる割り込み処理の設定は引き継ぎません。

- 読み込んだプログラムに、ユーザー定義関数の設定は引き継ぎません。
- 読み込んだプログラムでREAD文を実行すると、プログラムの最初のDATA文からデータを読み出します。
- OPTION BASE 文の設定は、無条件に読み込んだプログラムに引き継ぎます。
- RENUM コマンドで行番号を変更した場合、CHAIN 文中の行番号は変更しません。

C

参照　COMMON(引き渡す変数の指定)
LOAD(プログラムの読み込み)
MERGE(プログラムの連結)

プログラム例

メモリ上のプログラム

```
100 ' CHAIN
110 '----  CHAINを行い、変数を引き渡す  ----
120 '呼ぶ側のプログラム
130 A=2222
140 B=3333
150 C$="EPSON"
160 PRINT A,B,C$
170 PRINT
180 INPUT "プリンタに出力しますか?(Y/N)";D$
190 IF D$="Y" OR D$="y" THEN 210
200 IF D$="N" OR D$="n" THEN END
210 CHAIN "PRINT.BAS",ALL
220 END
```

ディスク上のプログラム(PRINT. BAS)

```
100 ' CHAIN
110 '呼ばれる側のプログラム
120 LPRINT A,B,C$
130 END
```

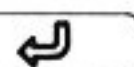
```
RUN
 2222          3333          EPSON

プリンタに出力しますか?(Y/N)?
 2222          3333          EPSON
```

CHR$

【キャラクタ・ダラー】

CHaRacter
文字(character)から文字コードを求めること。

機能　指定した数値(文字コード)に対応する文字を与えます。

書式　CHR$(文字コード)

使用例

PRINT CHR$(12)　←画面のデータを消去します。

KEY 1, "RUN"+CHR$(13)　←ファンクションキー1にRUN+CRコード(↵キー)を設定します。

解説

- 文字コードで指定した数値に対応する文字1字(1バイト文字)を与えます。文字コードは0から255の範囲の整数です。
- これとは逆に文字の文字コードを調べるには、ASC関数を用います。

参照

ASC(文字コードを調べる)
付録A　文字コード表

プログラム例

```
100 ' CHR$
110 ' --- 文字コードから文字に変換 ---
120 PRINT "  2 3 4 5 6 7 8 9 A B C D E F"
130 FOR I=0 TO 15
140   PRINT HEX$(I);" ";
150   FOR J=0 TO 13
160     PRINT CHR$(&H20+I+J*16);" ";
170   NEXT : PRINT
180 NEXT
190 END

RUN
  2 3 4 5 6 7 8 9 A B C D E F
0   0 @ P ` p ▁ ┴   ｰ ﾀ ﾐ ═ ╳
1 ! 1 A Q a q ▂ ┬ ｡ ｱ ﾁ ﾑ ╞ 円
2 " 2 B R b r ▃ ┤ ｢ ｲ ﾂ ﾒ ╪ 年
3 # 3 C S c s ▄ ├ ｣ ｳ ﾃ ﾓ ╡ 月
4 $ 4 D T d t ▅ ▔ ､ ｴ ﾄ ﾔ ◢ 日
5 % 5 E U e u ▆ ─ ･ ｵ ﾅ ﾕ ◣ 時
6 & 6 F V f v ▇ │ ｦ ｶ ﾆ ﾖ ◥ 分
7 ' 7 G W g w █ ▕ ｧ ｷ ﾇ ﾗ ◤ 秒
8 ( 8 H X h x ▏ ┌ ｨ ｸ ﾈ ﾘ ♠
9 ) 9 I Y i y ▎ ┐ ｩ ｹ ﾉ ﾙ ♡
A * : J Z j z ▍ └ ｪ ｺ ﾊ ﾚ ◇
B + ; K [ k { ▌ ┘ ｫ ｻ ﾋ ﾛ ♣
C , < L ¥ l ¦ ▋ ╭ ｬ ｼ ﾌ ﾜ ● ╲
D - = M ] m } ▊ ╮ ｭ ｽ ﾍ ﾝ ○
E . > N ^ n ~ ▉ ╰ ｮ ｾ ﾎ ﾞ ╱
F / ? O _ o   ┼ ╯ ｯ ｿ ﾏ ﾟ ╲
OK
```

CINT

【コンバート·インテジャ】

Convert INTeger
整数(integer)に変換(convert)すること。

機能　単精度実数や倍精度実数を四捨五入して、整数に変換します。

書式　CINT(数式)

使用例　PRINT CINT(X!*100); "%"　←X! の値を整数の百分率で表示します。

解説

- ( )内の**数式**の値を整数に変換します。
- 変換した結果の範囲は、整数の範囲(−32768〜32767)になります。これを超えた場合は、Overflow(桁あふれ)エラーになります。
- CINT 関数は、単精度実数や倍精度実数の小数第1位を四捨五入して整数型に変換します。これに対し、FIX 関数、INT 関数は数値の小数部を切り捨てて整数化します。型の変換や範囲チェックは行いません。
- 整数型変数に代入するときや、ステートメントや関数がパラメータとして整数型を要求しているときには自動的に整数への型変換を行います。A%=3042.12! と A%=CINT(3042.12!) の結果は同じになります。

参照

CDBL(倍精度実数に変換)
CSNG(単精度実数に変換)
FIX(小数部の切り捨て)
INT(小数部の切り捨て)

プログラム例

```
100 ' CINT / FIX / INT
110 ' --- CINT／FIX／INT関数の違い ---
120 PRINT "CINT／FIX／INT関数の違い"
130 INPUT "数値は :";A
140 PRINT "CINT=";CINT(A),"FIX=";FIX(A),"INT=";INT(A)
150 GOTO 130

RUN
CINT／FIX／INT関数の違い
数値は :? 12
CINT= 12       FIX= 12       INT= 12
数値は :? 12.5
CINT= 13       FIX= 12       INT= 12
数値は :? -12.5
CINT=-13       FIX=-12       INT=-13
数値は :?
```

CIRCLE

【サークル】

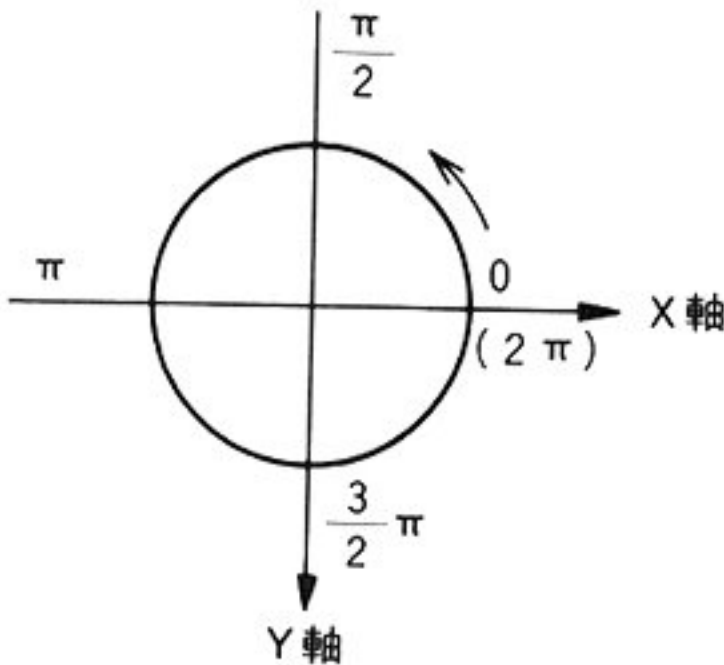
CIRCLE
円のこと。したがって円を描くこと。

機　能　位置や大きさ、形などを指定し、画面に円や楕円を描きます。

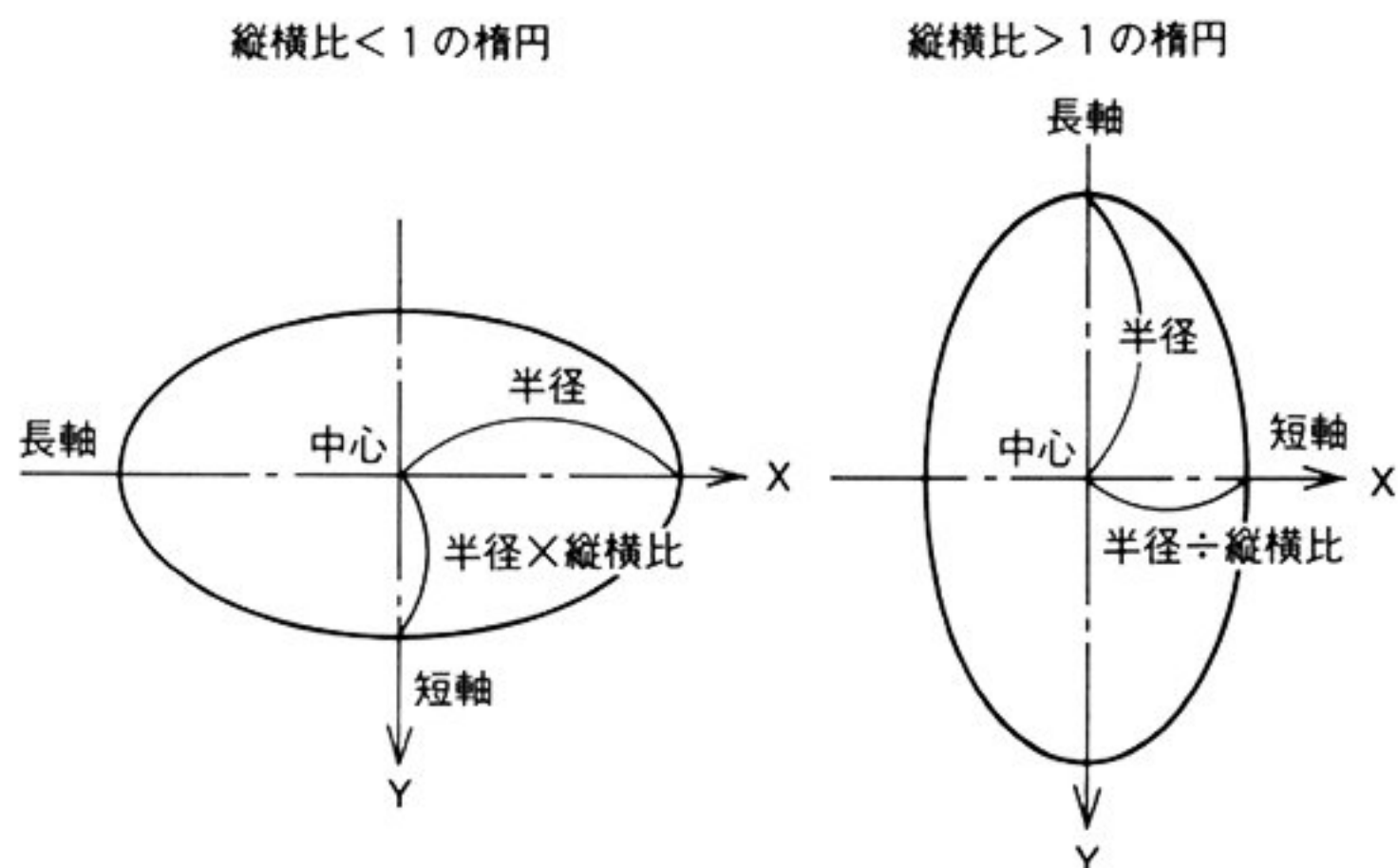
書　式　CIRCLE　(Wx, Wy), 半径 [, [描画色] [, [開始角度] [, [終了角度] [, [縦横比] [, F [, | 塗色 / タイルストリング |]]]]]]

使用例　CIRCLE(200, 100), 50　　←座標(200, 100)を中心として、半径50の円を描く。

解　説

・座標(**Wx, Wy**)で指定した点を中心として、**半径**の大きさの円を描きます。座標の指定は、ワールド座標の絶対座標か、最終参照座標からの相対座標で指定します。
・**半径**は、円の場合は半径の長さ、楕円の場合は長軸の半分の長さになります。楕円の短軸の長さは**縦横比**から決まります。
・**描画色**は、カラーモードのときに有効で、線の色をパレット番号で指定します。省略した場合はCOLOR文で指定した前景色になります。
・**開始角度、終了角度**は、円の描き始めと描き終わりの角度をラジアンで指定します。指定できる範囲は－2πから2π(π=3.141593)になります。**開始角度**を省略すると0を、**終了角度**を省略すると2πを指定したことになります。

・**開始角度、終了角度**に負の値を指定すると、開始点または終了点を円の中心と結び、扇形を描くことができます。このとき、開始角度、終了角度の値は絶対値を用います。

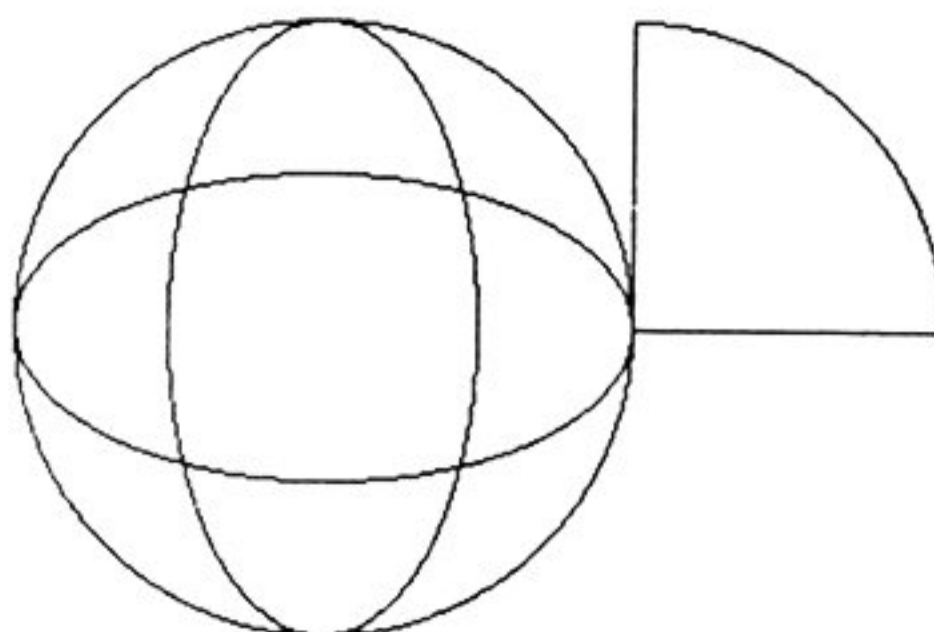
• **縦横比**により楕円を描くことができます。**縦横比**は、縦/横の比率を数値(数式)で指定したものです。**縦横比**が1より小さい場合は、X軸方向が長軸となり、1より大きい場合は、Y軸方向が長軸になります。

縦横比を省略すると、解像度が640×200ドットのディスプレイでは0.5を、640×400ドットのディスプレイでは1を指定したことになります。

• **F**を指定をすると、描いた円の内部を**塗色**で指定したパレット番号の色、または**タイルストリング**で指定した模様で塗りつぶします。**塗色**および**タイルストリング**を省略すると、円を描いた色で塗りつぶします。

• CIRCLE文を実行後、最終参照座標は円の中心に移ります。

参　照　　PAINT(領域内部の塗りつぶし)

プログラム例

```
100 ' CIRCLE
110 ' ---  円を描く  ---
120 SCREEN 2
130 CLS 3
140 PI=3.14159
150 CIRCLE(200,100),100
160 CIRCLE(200,100),100,,,,.5
170 CIRCLE(200,100),100,,,,2
180 CIRCLE STEP(100,0),100,,-.00001,-PI/2
190 END
OK
```

CLEAR

【クリア】

CLEAR
一掃する、取り除くという意味から変数を初期化すること。

機能　すべての変数領域を初期化します。同時に機械語プログラム領域、スタック領域、配列変数領域の大きさを設定します。

書式　CLEAR　[[ダミー] [, [プログラム領域] [, [スタック領域] [, [配列変数領域]]]]]

使用例　CLEAR　←すべての変数を初期化します。

解説

- CLEAR 文は、BASIC プログラム中で使用した変数の内容をすべて次のように初期化します。ただしプログラムは消去しません。

  数値変数……………………………………0
  文字変数……………………………………空文字列(ヌルストリング)
  配列の定義(DIM文)………………………無効
  配列の規定(OPTION BASE 文)…………0
  変数の型宣言………………………………無効
  ユーザー定義関数…………………………無効

  配列変数だけを初期化するには ERASE 文を用います。
- **ダミー**は、どんな値を指定しても意味を持ちません。省略してください。
- **プログラム領域**は、BASIC インタープリタの作業領域とプログラムやデータを記憶する領域の大きさをセグメントアドレスで指定します。このアドレス以降は BASIC インタープリタの使用範囲外になり、機械語プログラムなどのデータを記憶することができます。プログラム領域を省略すると、最大限の**プログラム領域**を確保します。
- **スタック領域**は、BASIC プログラムで使用するスタックの大きさをバイト数で設定します。このスタックはプログラム中の FOR～NEXT 文のループや GOSUB 文の多重化(ネスティング)、複雑なグラフィックの描画などで使用します。プログラム実行中に Out of memory(メモリ容量不足)エラーが発生したときなどに、この領域を大きくするとエラーが発生しなくなることがあります。**スタック領域**を省略すると、512バイトをスタックとして設定します。
- **配列変数領域**は、数値型配列のデータを記憶する領域の大きさを設定します。ここで指定した値の16倍の大きさ(バイト数)が変数領域になります。

参照　ERASE(配列変数の消去)
NEW(プログラムの消去)

CLOSE

【クローズ】

CLOSE
閉じるという意味から、ファイルを閉じて入出力処理を終了すること。

機能　指定したファイルの入出力処理を終了します。

書式　CLOSE　[[#] ファイル番号 [, [#] ファイル番号] ・・・]

使用例　CLOSE #1　←ファイル番号1のファイルをクローズします。

解説
- **ファイル番号**で指定したファイルの終了処理(クローズ)を行います。カンマ(,)で区切って並べて、複数のファイルを指定することができます。
- **ファイル番号**を省略すると、その時点でオープンしているすべてのファイルをクローズします。また、この場合はハードディスクドライブの磁気ヘッドのリトラクトが行われます。
- クローズすると、ファイルとファイル番号の関係が切り離され、そのファイル番号を別のファイルのファイル番号として指定できます。
- ファイルを出力モードでオープンしていた場合は、ファイルバッファに残っているデータをファイルに出力します。
- CLOSE文のほか、END, NEW文およびRUN, LOAD文でもファイルをクローズします。
- STOP文あるいはEND文なしでプログラムの最終行に至り、プログラムの実行が終了した場合は、ファイルのクローズは行いません。

参照　OPEN(ファイルのオープン)

CLS

【シー・エル・エス/クリア・スクリーン】

CLear Screen
画面(screen)を消去(clear)すること。

機能　画面の文字やグラフィックを消去します。

書式　CLS　[機能]

使用例　CLS 3　←テキスト画面とグラフィック画面を消去します。

解説
- **機能**で指定した画面を消去します。
- **機能**は1から3までの数値を指定します。省略した場合は1を指定したことになります。

機能	意味
1 (省略時)	テキスト画面を消去します。テキスト画面が白黒モードで文字がリバースモードになっている場合は、画面は白くなります。
2	グラフィック画面を消去します。カラーモードの場合は、背景色で消去します。
3	テキスト画面とグラフィック画面を消去します。

- テキスト画面の消去範囲は、CONSOLE文で設定したスクロール範囲内です。
- グラフィック画面の消去範囲は、VIEW文で設定したビューポート範囲内です。
- グラフィック画面の消去を行った場合、最終参照座標はビューポートの左上の点に移ります。
- CLR キーはテキスト画面を消去します。

COLOR[1]

【カラー】

COLOR
色の意味。

機　能　表示する文字の色、画面の色、グラフィック画面で使用できる色の設定を行います。

書　式　COLOR　[[文字の色] [, [背景色] [, [境界色] [, [前景色] [, [カラーモード]]]]]]

使用例　COLOR 4,7,,5　←文字の色を緑、背景色を白、グラフィックの描画色を水色にします。

解　説　• **文字の色**は、テキスト画面に表示する文字の色です。テキスト画面がカラーモードか白黒モードかにより機能が異なります。

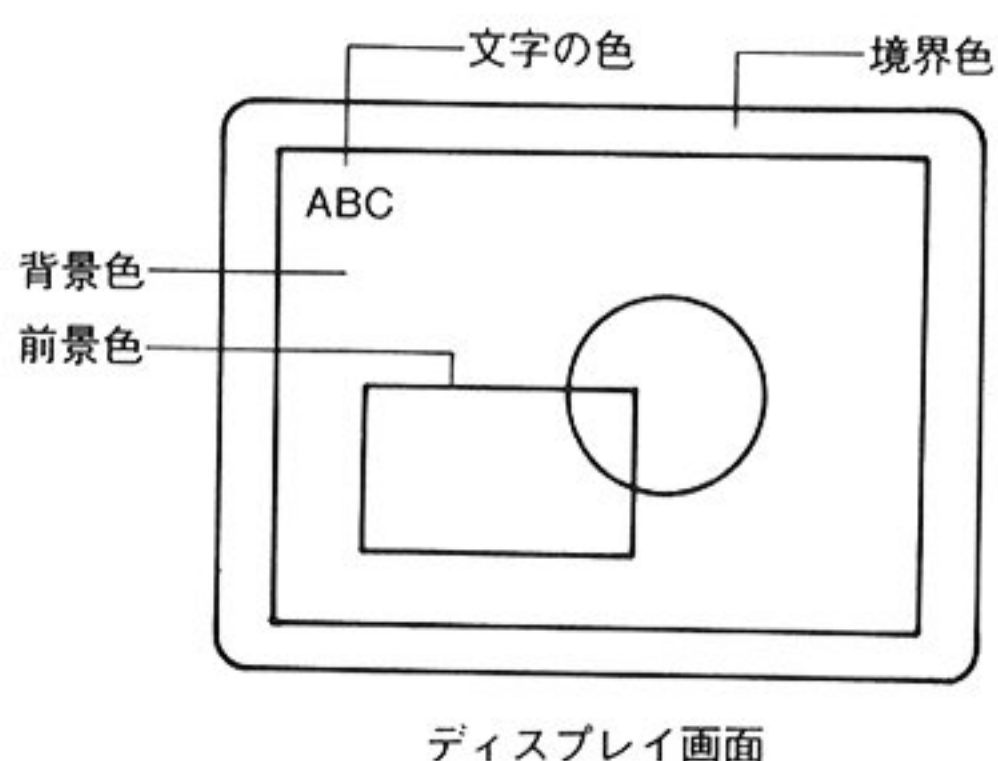
文字の色	カラーモード(文字の色)	白黒モード(文字の機能)
0	黒	ノーマル(通常の表示)
1	青	ヒドン(表示しない)
2	赤	ブリンク(点滅表示)
3	紫	ヒドン
4	緑	リバース(反転表示)
5	水色	リバースヒドン(反転無表示)
6	黄色	リバースブリンク(反転点滅表示)
7	白	リバースヒドン

カラーモード、白黒モードの指定はCONSOLE文で行います。テキスト画面の文字の色は、この文字の色によって決定し、COLOR文で指定するパレット番号とは無関係です。

• **背景色**(バックグランドカラー)にはパレット番号で、グラフィック画面の背景の色を指定します。CLS文でグラフィック画面の消去をすると、この色で画面を塗りつぶします。

• **境界色**(ボーダーカラー)は標準ディスプレイを接続しているときだけ意味を持ちます。ディスプレイの全領域のうち、BASICの使用しない領域の色を、カラーコード(パレット番号ではない)で指定します。高解像ディスプレイを接続しているときは省略します。

• **前景色**（フォアグランドカラー）にはパレット番号で、グラフィック画面に点や線を描くときに用いる色を指定します。グラフィック関係の命令を実行する際、色の指定を省略すると、前景色で描画します。

ディスプレイ画面

• **カラーモード**は、グラフィック画面で表示する色の種類を、0から2の数値で指定します。グラフィック画面をカラーモードにするには、SCREEN 文を実行します。また、4096色中8色モード、または4096色中16色モードを使うためには本体のディップスイッチ SW1-8をONにしてから BASIC を起動します。

カラーモード	意味
0	8色中8色モード　（8個のパレットに8色を対応づけるモード）
1	4096色中8色モード（8個のパレットに4096色中の8色を対応づけるモード）
2	4096色中16色モード（16個のパレットに4096色中の16色を対応づけるモード）

各モードにおけるパレットとカラーコードの対応づけは、COLOR 文で行います。またカラーモードの設定が切り替わると、カラーコードとパレットの対応づけは初期化されます。初期設定については COLOR[2] 文を参照してください。

参　照

COLOR（カラーパレットの指定）
COLOR@（表示文字の色の変更）
CONSOLE（テキスト画面のモード設定）
SCREEN（グラフィック画面のモード設定）

COLOR²

【カラー】
COLOR
色の意味。

C

機能 カラーコードとパレットの対応づけを行います。

書式 COLOR [=(パレット番号, カラーコード)]

使用例

COLOR	←カラーコードとパレットの対応づけを初期化します。
COLOR=(2,4)	←パレット2に4(緑)を対応づけます。

解説

- 指定した**パレット番号**に、指定した**カラーコード**を対応づけます。グラフィック関係の命令で描画色の指定を行う場合は、直接カラーコードを指定するのではなくパレット番号を指定することになります。
- **パレット番号**は、パレットに付けた番号で、カラーモードの違いにより指定できる範囲が異なります。

カラーモード	指定範囲
8色モード	0から7
4096色中8色モード	0から7
4096色中16色モード	0から15

- **カラーコード**は、実際に表示する色を指定するもので、8色モードと4096色モードで指定方法が異なります。

8色モード

カラーコード	0	1	2	3	4	5	6	7
色	黒	青	赤	紫	緑	水色	黄色	白

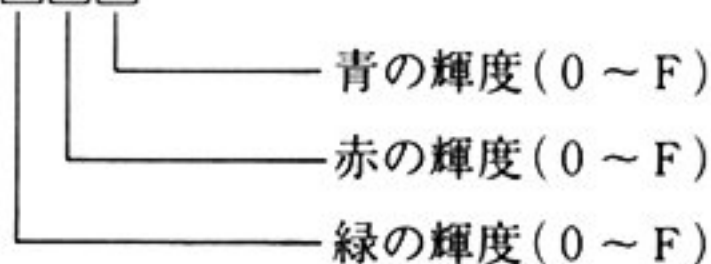
4096色中8色モード・4096色中16色モード

カラーコードを、光の3原色(赤・緑・青)の混ぜ合わせる量(輝度)によって表現します。赤、緑、青は、それぞれ16段階の輝度を表現するすることができ、3色、16段階の輝度を組み合わせることによって4096色を表現することができます。

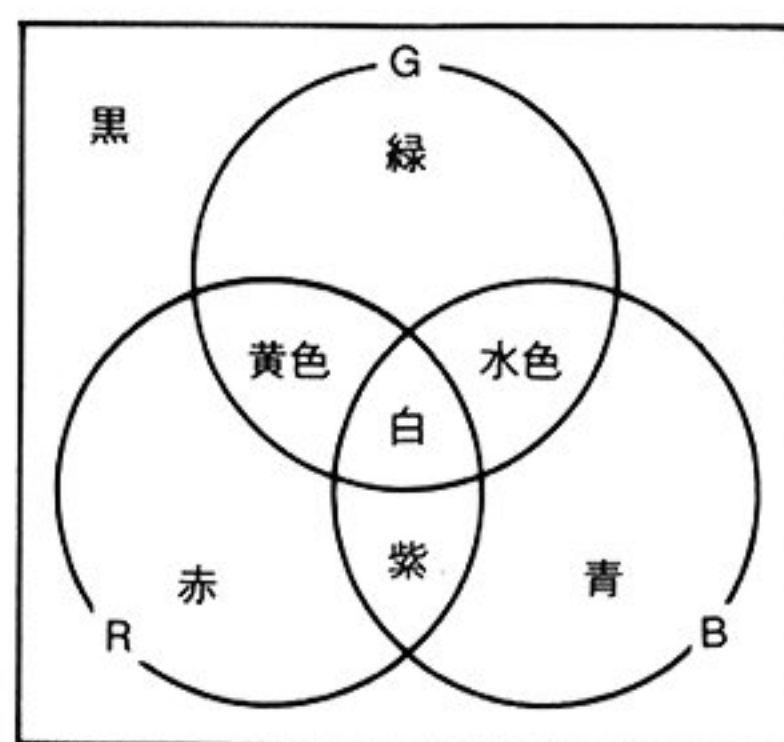
(赤の16輝度)×(緑の16輝度)×(青の16輝度)=4096色

カラーコードは0からFまでの16進数3桁で表します。

カラーコード=&h□□□

- 1桁目: 緑の輝度(0~F)
- 2桁目: 赤の輝度(0~F)
- 3桁目: 青の輝度(0~F)

0はその色がないことを意味し、Fは最も明るい色になります。

輝度	0 ・・・・・・・・・・・・・・・・・・・・・・・・・・・・ F
	暗い 明るい

赤、緑、青の組み合わせで表現される基本的な色の関係は次のようになります。

＊カラーコードは10進数で表現することもできますが、16進数で表したほうがわかりやすいくなります。

・**パレット番号、カラーコード**を省略すると、パレット番号、カラーコードの関係を次のように初期化します(BASIC 起動時の設定)。

画面モード／パレット番号	8色モード	4096色中・8色モード	4096色中・16色モード	
0	0 (黒)	&H000(黒)	&H000(黒)	明るい
1	1 (青)	&H00F(青)	&H00F(青)	
2	2 (赤)	&H0F0(赤)	&H0F0(赤)	
3	3 (紫)	&H0FF(紫)	&H0FF(紫)	
4	4 (緑)	&HF00(緑)	&HF00(緑)	
5	5 (水色)	&HF0F(水色)	&HF0F(水色)	
6	6 (黄色)	&HFF0(黄色)	&HFF0(黄色)	
7	7 (白)	&HFFF(白)	&HFFF(白)	
8	－	－	&H777(灰色)	暗い
9	－	－	&H00A(青)	
10	－	－	&H0A0(赤)	
11	－	－	&H0AA(紫)	
12	－	－	&HA00(緑)	
13	－	－	&HA0A(水色)	
14	－	－	&HAA0(黄色)	
15	－	－	&HAAA(白)	

－　は指定不可

〔参　照〕　COLOR[1](カラーモードの指定)

SCRREN(カラーモードの設定)

注意：4096色中8色モードおよび4096色中16色モードでアナログ RGB ディスプレイを接続していない場合には指定どおりの色は表示できません。

COLOR@

【カラー・アットマーク】

COLOR@
画面上の文字に色や機能を付けること。

C

機　能　すでにテキスト画面に表示している文字の色を変えたり、表示状態を変えます。

書　式　COLOR@(X1, Y1)－(X2, Y2) [, 文字の色]

使用例　COLOR@(0, 0)－(39, 12), 4　←画面の上から13行目の40桁までに表示している文字を緑色に変えます。

解　説

- キャラクタ座標で指定した2点**(X1, Y1)**、**(X2, Y2)**を対角とする四角形内に表示している文字の状態を、**文字の色**で指定した状態に変えます。
- 指定した領域内に、何も表示がない場合は何の効果もありません。
- **文字の色**の機能は、テキスト画面のモード(カラーモードか白黒モードか)によって異なります。それぞれのモードにおける機能は、COLOR[1] 文を参照してください。**文字の色**を省略した場合は7を指定したことになります。
- COLOR@ 文実行後に、指定範囲内に新しく書いた文字は何の影響も受けません。直前に実行したCOLOR文で指定した文字の色で表示します。

参　照

COLOR[1](文字の色の指定)
CONSOLE(テキスト画面のモード指定)

プログラム例

```
100 ' COLOR@
110 ' --- 文字に色を塗る ---
120 CONSOLE ,,,1
130 CLS
140 FOR I=0 TO 16
150  PRINT "PERSONAL COMPUTER"
160 NEXT
170 C=0
180 FOR  I=0 TO 16
190 C=C MOD 8
200   COLOR@(0,0)-(I,I),C
210   C=C+1   : FOR W=1 TO 100 : NEXT
220 NEXT : GOTO 180
```

COM ON
COM OFF
COM STOP

【コム・オン】
【コム・オフ】
【コム・ストップ】

COMmunication
コミュニケーションは通信を意味する。これから通信に関する制御を行うこと。

機能　通信回線による割り込み処理ルーチンの実行を許可、禁止、停止します。

書式　COM[(回線番号)] |ON / OFF / STOP|

使用例
```
COM ON        ←割り込み処理ルーチンを実行します。
COM OFF       ←割り込み処理ルーチンの実行を禁止します。
COM STOP      ←割り込み処理ルーチンの実行を停止します。
```

解説
- 通信回線よりデータを受信したときに実行する割り込み処理ルーチンの制御を行います。この命令を実行する前に通信回線をオープンしていなければいけません。
- 割り込み処理ルーチンは ON COM GOSUB 文で定義します
- **回線番号**は次の数値で指定し、省略した場合はすべての通信回線が対象になります。

回線番号	通信回線
1	内蔵 RS-232C インターフェイス
2	拡張 RS-232C インターフェイス 1
3	拡張 RS-232C インターフェイス 2

- **COM ON** は割り込み処理ルーチンの実行を許可します。この命令を実行後は、データを受信すると ON COM GOSUB 文で定義した割り込み処理ルーチンを実行します。
- **COM OFF** は割り込み処理ルーチンの実行を禁止します。この命令を実行後は、データを受信しても割り込み処理ルーチンを実行しません。また受信データの受信バッファへの記憶も行いません。
- **COM STOP** は割り込み処理ルーチンの実行を停止します。この命令を実行後は、データを受信しても割り込み処理ルーチンは実行しません。ただし、受信データは記憶しており、その後 COM ON を実行するとただちに割り込み処理ルーチンを実行します。
- プログラムの実行を終了するときは、COM OFF を実行してください。
- ON COM GOSUB 文で指定した割り込み処理ルーチン実行中は、自動的に割り込み停止状態になります。

参照　ON COM GOSUB(通信回線による割り込み処理ルーチンの定義)

COMMON

【コモン】

COMMON
共同の、公共のという意味から、異なったプログラムの間に、共通の変数を持つこと。

機　能　実行中のプログラムから、CHAIN 文で実行するプログラムに変数を引き渡します。

書　式　COMMON　変数名 [[, 変数名] ・・・]

使用例

```
100  COMMON A, B$, C( )
        :
2560  CHAIN "PROG"
```

← 変数 A, B$, 配列 C の値をそのまま、プログラム PROG に渡します。

解　説

- **変数名**に指定した変数を、CHAIN 文で実行するプログラムに引き渡します。変数は、カンマ(,)で区切って必要な数だけ並べます。
- 配列変数を指定するときは変数名の後ろにかっこ( )を付け加えます。また、引き渡すプログラム中で配列の宣言(DIM 文)を行う必要はありません。
- COMMON 文は非実行文であり、プログラム中のどの行に置いても構いません。しかし、プログラムの最初に記述した方がわかりやすくなります。
- COMMON 文は、一つのプログラム中でいくつでも使うことができますが、変数名を重複して指定することはできません。
- COMMON 文で指定した変数に一度も代入を行わないと CHAIN 文を実行中に Illegal function call(違法関数呼び出し)エラーになります。
- COMMON 文は変数を一つずつ指定しますが、CHAIN 文で ALL 指定をすれば、すべての変数を引き渡すことができます。

参　照　CHAIN(プログラムを読み込んで実行)

CONSOLE

【コンソール】

CONSOLE
制御装置の意味から転じてテキスト画面のモードを設定すること。

機 能　テキスト画面の種々のモードを設定します。

書 式　CONSOLE　[スクロール開始行] [, [スクロール行数] [, [ファンクションキー表示] [, [表示モード]]]]

使用例　CONSOLE 0,24, 1 , 0　←画面の全領域をスクロール範囲とし、ファンクションキーを表示し、白黒モードに設定します。

解 説
- **スクロール開始行、スクロール行数**は、画面のスクロール範囲を設定します。スクロール開始行はキャラクタ座標の行の位置を指定します。CLS 文のテキスト画面のクリアは、この範囲内を消去します。
- **ファンクションキー表示**は、画面の最下行のファンクションキーの表示の有無を指定します。

0	ファンクションキーを表示しない。
1	ファンクションキーを表示する。

- **表示モード**はテキスト画面の白黒/カラーモードの指定を行います。

0	白黒モード
1	カラーモード

- BASIC 起動直後の設定は次のようになっています。

参 照　COLOR[1] (表示する文字の色)
SCREEN (グラフィック画面のモード設定)

CONT

【コント/コンティニュー】

CONTinue
続ける、継続するの意味からプログラムの実行を継続すること。

機能　[STOP]キーを押したりSTOP文を実行して中断したプログラムの、実行を再開します。

書式　CONT

使用例　CONT　←中断していたプログラムの実行を再開します。

解説
- プログラムの実行を[STOP]キー([CTRL]+[C])の入力、あるいはSTOP文により中断した後、ダイレクトモードで**CONT**を入力すると実行を再開することができます。
- プログラムの実行は、中断した行の次の行から再開します。INPUT文実行中に中断した場合は、INPUT文の実行から再開します。
- プログラムの実行を中断して、ダイレクトモードで変数の値を調べたり、変更したりすることができます。
- ただし、中断している間にプログラムを修正すると、CONTコマンドを実行しても、Can't continue(継続不可能)エラーとなり、実行を再開することはできません。

参照　END(プログラムの実行の終了)
STOP(プログラムの実行の停止)

COPY

【コピー】

COPY
複写の意味。画面のハードコピーを行うこと。

機能 画面のハードコピー(画面の表示をそのままプリンタに出力)を行います。

書式 COPY [機能]

使用例

```
COPY 2
```

←グラフィック画面を出力します。

解説

- **機能**に1から5までの数値を指定し、画面のハードコピーの範囲を指定します。この機能の内容は、メモリスイッチ SW6-4の設定(画面ハードコピー機能の拡張)によって異なります。
- **機能**を省略すると3を指定したことになります。
- **メモリスイッチSW6-4が0の場合**

機能	意味
1	テキスト画面のみのハードコピーを行います。
2	グラフィック画面のみのハードコピーを行います。
3	テキスト画面とグラフィック画面を重ね合わせてハードコピーを行います。テキスト画面の文字は、プリンタの印字体になります。
4	標準モードの場合にグラフィック場面を縦方向に縮小してハードコピーを行います。
5	標準モードの場合にグラフィック画面を縦方向に縮小して、テキスト画面と重ね合わせてハードコピーを行います。

高解像モード時に機能4または5を指定した場合はそれぞれ機能2と3と同じになります。

- **メモリスイッチSW6-4が1(画面ハードコピー機能の拡張)の場合**

機能	意味
1	テキスト画面のみのハードコピーを行います。
2	グラフィック画面のみを出力します。カラーコピーの場合は、白黒を反転してハードコピーを行います。
3	テキスト画面とグラフィック画面を重ね合わせてハードコピーを行います。テキスト画面の文字は、ビットイメージで出力します。* カラーコピーの場合は、白黒を反転して出力します。
4	グラフィック画面のみのハードコピーを行います。白黒を反転しません。
5	テキスト画面とグラフィック画面を重ね合わせてハードコピーを行います。白黒を反転しません。テキスト画面の文字は、ビットイメージで出力します。*

*テキスト画面が20行モードで文字を反転表示している場合、画面のイメージと異なります。

カラーコピーはメモリスイッチSW5-3を1にします。ただし4096色中8色モードおよび16色モードのときはカラーコピーを指定しても白黒コピーになります。

参照 日本語Disk BASICユーザーズマニュアル 第7章 プリンタ

COS

【コサイン】

COS
三角関数の cos(余弦)の意味。

C

機能　余弦(コサイン)を計算します。

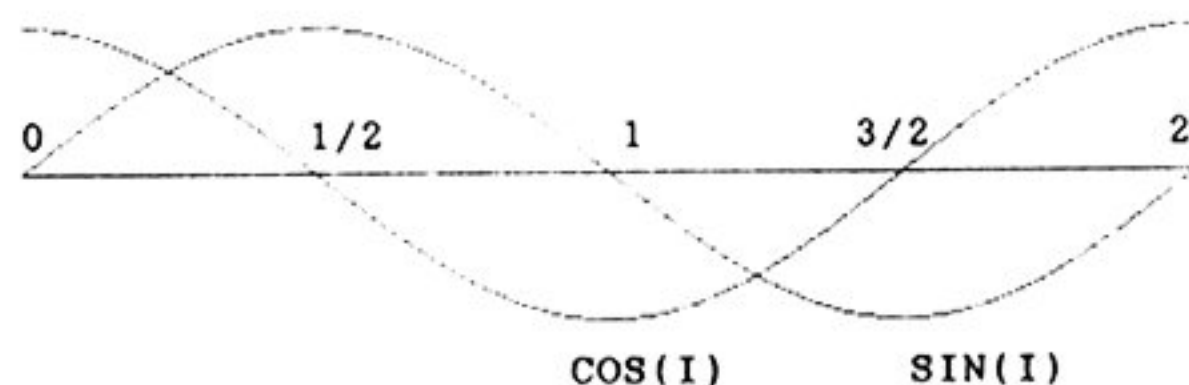
書式　COS(数式)

使用例
PI#=3.14159265358979323846　←30°の余弦を計算します。
A=COS(30*PI#/180)

解説
• (　)内の**数式**の余弦を計算します。数値の単位はラジアンです。角度が度で与えられている場合は、ラジアンに変換するためにπ/180をかけます。
πには次のような値を使用すると最も精度の高い計算を行うことができます。
倍精度実数(PI#をπの値を持つ変数とします)
PI#=3.14159265358979323846
• **数式**が倍精度実数のときは、結果も倍精度で計算した値になります。他の場合は、単精度で計算した値になります。

参照
ATN(逆正接)
SIN(正弦)
TAN(正接)

プログラム例

```
100 ' COS / SIN
110 ' --- SIN, COSカーブを描く ---
120 SCREEN 2
130 CLS 3
140 PI=3.14159 : D=PI/180
150 LINE (0,50)-(360,50)
160 LOCATE 0,2
170 PRINT "0           1/2           1           3/2           2"
180 FOR I=0 TO 360
190   S=SIN(I*D) : C=COS(I*D)
200   SY=CINT(50-S*50)
210   CY=CINT(50-C*50)
220   PSET(I,SY) : PSET(I,CY)
230 NEXT
```

CSNG

【コンバート・シングル】

Convert SiNGle
単一のという意味の single から単精度、すなわち単精度に変換(convert)すること。

機能　整数、倍精度実数を単精度実数に変換します。

書式　CSNG(数式)

使用例　A$=STR$(CSNG (A#))　←A#の値を単精度の桁数で文字列にします。

解説

- ( )内の**数式**の値を単精度実数に変換します。このとき、有効桁数を超えている部分は丸めます。また変換の結果、単精度実数で表わすことのできる範囲を超えた場合はOV(桁あふれ)エラーになります。
- CSNG 関数は、数値を有効桁が2進数で24桁の単精度実数に変換します。
- 単精度変数へ代入するときや、ステートメントや関数がパラメータとして単精度実数を要求しているときには自動的に単精度への型変換を行います。**A!=3042.1545452#** と**A!=CSNG(3042.1545452#)**の結果は同じになります。

参照　CINT(整数に変換)
CDBL(倍精度実数に変換)

CSRLIN

【カーソル・ライン】

CurSoR LINe
line は行のこと。したがってカーソル(cursor)の行位置。

機能 画面上のカーソルの行位置をキャラクタ座標で与えます。

書式 CSRLIN

使用例 A=CSRLIN ←Aに現在のカーソルのあるY座標の値を代入します。

解説
- カーソルのある行をキャラクタ座標のY座標で与えます。
- カーソルの桁位置を知るには、POS関数を用います。

参照 POS(カーソルの桁位置を調べる)

プログラム例

```
100 ' CSRLIN
110 '----  カーソルの位置を知る  ----
120 CONSOLE 0,25,1,0 : CLS
130 FOR Y=4 TO 14
140   GOSUB *ART
150 NEXT Y
160 FOR Y=14 TO 4 STEP -1
170  GOSUB *ART
180 NEXT Y
190 GOTO 130
200 *ART
210  LOCATE 10,Y : PRINT "EPSON"
220  L=CSRLIN
230  LOCATE 40,10 :PRINT "CURSOR_LINE == 10 ,";L
240  FOR I=0 TO 300 : NEXT I
250  LOCATE 10,Y : PRINT "        "
260 RETURN
```

CVI
CVS
CVD

【シー・ブイ・アイ】
【シー・ブイ・エス】
【シー・ブイ・ディ】

ConVert to Integer/Single/Double
それぞれ整数(integer)、単精度実数(single)、倍精度実数(double)に変換(convert)します。

機能　数値の内部コードの形式を持った文字列を、実際の数値データに変換します。

書式
CVI(2バイトの文字列)
CVS(4バイトの文字列)
CVD(8バイトの文字列)

使用例

```
A%=CVI(I$)        ←I$を整数に変換
B!=CVS(S$)        ←S$を単精度実数に変換
C#=CVD(D$)        ←D$を倍精度実数に変換
```

解説
• MKI$、MKS$、MKD$関数によって、内部形式の文字列に変換した数値を、もとの数値扱いのデータに変換します。

数値の型	数値→文字列の関数	文字列→数値の関数	文字列の長さ
整数	MK I$	CVI	2バイト
単精度実数	MK S$	CVS	4バイト
倍精度実数	MK D$	CVD	8バイト

参照　MKI$/MKS$/MKD$(数値から文字列への変換)
日本語Disk BASICユーザーズマニュアル　第11章　データの内部構造

DATA

【データ】

DATA
データという意味からプログラム中にデータを定義すること。

機能 READ文で読み出す、文字データ・数値データを定義します。

書式 DATA 定数 [, 定数] ・・・

使用例 DATA ABC, 100, DEF, 200 ←READ文でABCから順に読み出します。

解説

- DATA文に続けて、**数値定数・文字定数**をカンマ(,)で区切って並べます。
- **数値定数**は、整数(10進数表記、16進数表記、8進数表記)、固定小数点表記、指数小数点表記のどの形式でも使用できます。ただし、10*20などの定数式は使用できません。
- **文字定数**に、カンマ(,)、コロン(:)や先頭に意味のある空白がある場合は、文字定数を二重引用符(")で囲んで指定します。
- READ文は、DATA文で定義した定数を順次読み出します。このとき、数値変数に文字データを読み込もうとすると、Syntax error(構文の誤り)エラーになります。文字変数に数値定数を読み込むと、読み込んだデータは文字データとして扱われます。
- READ文で読み出すDATA文の行は、RESTORE文で指定します。RESTORE文の指定がない場合、プログラムの最初に現れるデータから読み出します。

参照 READ(データの読み出し)
RESTORE(読みだすDATA文の行の指定)

プログラム例

```
100 ' DATA / READ / RESTORE
110 ' ---  データを定義して読み出す ---
120 RESTORE 230
130 READ M$,D$,Y$
140 PRINT M$,D$,Y$
150 '
160 RESTORE
170 READ A,B,C,D,E,F
180 PRINT A,B,C
190 PRINT D,E,F
200 END
210 DATA 06,05,1987
220 DATA 07,10,1987
230 DATA 月,日,年

RUN
月             日             年
 6             5              1987
 7             10             1987
OK
```

DATE$

【デート・ダラー】

DATE
日付のこと。

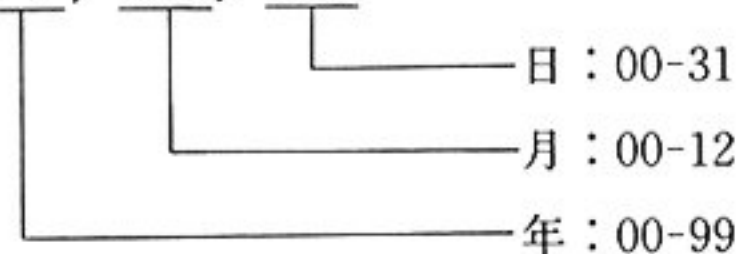
機能　日付を設定したり、現在の日付の読み出したりします。

書式
DATE$
DATE$="年/月/日"

使用例
PRINT DATE$　←現在の日付を表示します。
DATE$="87/12/19"　←現在の日付を設定します。

解説
- カレンダ時計に次の形式で**日付**を設定します。

DATE$="YY/MM/DD"

DD ── 日：00-31
MM ── 月：00-12
YY ── 年：00-99

年は西暦の下二桁です。月、日の設定で1から9までの数値の場合は、0を補って、01から09の形式で指定します。

- カレンダ時計の日付を**8文字の文字列**として与えます。与えられる文字列の形式は、日付を設定する形式と同じです。

参照　TIME$(時刻の設定・読み出し)

プログラム例

```
100 ' DATE$ / TIME$
110 ' --- 日付と時刻を設定する ---
120 PRINT "今日の日付は";DATE$;"です。"
130 INPUT "日付を入力してください。YY/MM/DD ";DT$
140 DATE$=DT$
150 PRINT "現在の時刻は";TIME$;"です。"
160 INPUT "時刻を入力してください。HH:MM:SS ";TM$
170 TIME$=TM$
```

DEF FN

【デファイン・ファンクション】

DEFine FuNction
関数(function)を定義(define)すること。

D

機 能　ユーザー関数を定義します。

書 式　DEF　FN名前［(引数［, 引数］・・・)］＝関数の定義式

使用例

100　DEF FNTRI(X, Y)=X*Y/2　　←関数FNTRI(X, Y)＝X*Y/2を定義します。

200　PRINT FNTRI(12.5, 7.3)　　←実際に $\frac{12.5\times7.3}{2}$ を計算します。

解 説

- **FN名前**という関数を定義します。

```
  変数名
   ┌─┐
FNTRI
└───┘
 関数名
```

- **引数**は、関数の定義式の中でのみ有効な変数名です。プログラム中に同じ名前の変数を使用しても影響を及ぼしません。
- **関数の定義式**は、その関数の演算の内容を記述した式で、プログラムの1行の範囲内で記述します。**引数**で指定した変数は、関数の定義式の中の変数に対応します。関数の定義式中のそれ以外の変数は、プログラム中の変数が対応します。
- 関数は数値関数、文字関数いずれも定義できますが、**名前**で指定した変数の型と関数の定義式で計算した結果の型は一致していなければなりません。
- BASICはFNで始まる変数はDEF FN文で定義した関数とみなします。したがってFNで始まる文字列を変数名としては使用できません。
- 複雑な関数式を定義すると、実行時にOut of memory(メモリが不足)エラーになることがあります。

プログラム例

```
100 ' DEF FN
110 '---  三角形の面積  ---
120 '
130 DEF FNTRI(X,Y)=X*Y/2
140 INPUT "底辺 = ";W
150 INPUT "高さ = ";H
160 PRINT : PRINT "面積は ";FNTRI(W,H)
170 END

RUN
底辺 = ? 12.5
高さ = ? 7.4

面積は  46.25
OK
```

DEFINT
DEFSNG
DEFDBL
DEFSTR

【デファイン・インテジャ】
【デファイン・シングル】
【デファイン・ダブル】
【デファイン・ストリング】

DEFine INTeger/SiNGle/DouBLe/STRing
整数(integer)、単精度実数(single)、倍精度実数(double)、文字(string)の型を宣言(define)する。

機能　変数の型を宣言します。

書式

DEFINT DEFSNG DEFDBL DEFSTR	英字	[, 英字] ・・・ ー英字

使用例

```
DEFINT A, Z
```
←A, Zで始まる変数名を持つ変数は整数型変数です。

```
DEFSNG A－Z
```
←すべての変数は単精度型変数です。

解説

- 指定した**英字**で始まる変数名を持つ変数を、**指定した型**に設定します。

  整数型　：DEFINT
  単精度型：DEFSNG
  倍精度型：DEFDBL
  文字型　：DEFSTR

- **英字**は、AからZまでのアルファベットから1文字を指定します。**英字**をカンマ(,)で区切って並べた場合は、その並べた**英字**を指定したことになります。またハイフン(-)でつなげると、その間のすべての**英字**を指定したことになります。
- これらの型宣言より、型宣言文字(%、!、#、$)による型宣言が優先します。また、いずれの型宣言も行っていない変数は単精度型になります。

参照　%、!、#、$(型宣言文字)

DEF SEG

【デファイン・セグメント】

DEFine SEGment
セグメント(segment)を定義(define)します。

機　能　直接メモリのアドレスを指定する命令の、セグメントアドレスを定義します。

書　式　DEF SEG ［＝アドレス］

使用例　DEF SEG＝&H8000　　←セグメントアドレスを &H8000 に設定します。

解　説

• 直接メモリのアドレスを指定する命令の、**セグメントアドレス**を定義します。このような命令には次のものがあります。

BLOAD(機械語プログラムのロード)
BSAVE(機械語プログラムのセーブ)
CALL(機械語プログラムの実行)
DEF USR(機械語プログラムの定義)
MON(モニタ機能)
PEEK(メモリから1バイト読み出し)
POKE(メモリへ1バイト書き込み)

これらの命令で指定するアドレスは、DEF SEG 文で定義した**アドレス**(セグメントアドレス)のオフセットアドレスになります。実際のアドレスは次のようになります。

例　**DEF SEG＝&H8000**　…セグメントアドレス　　8 0 0 0
　　PRINT HEX $(PEEK(&H1234))…オフセットアドレス　＋　1 2 3 4
　　　　　　　　　　実際のアドレス　　8 1 2 3 4

PEEK 関数で読み出すメモリのアドレスは &H81234番地になります。

• **DEF** と **SEG** の間には空白を1つ入れます。

• **アドレス**を省略すると CLEAR 文で設定するプログラム領域で指定したセグメントアドレスを指定したことになります。

DEF USR

【デファイン・ユーザー】

DEFine USeR
ユーザーが定義すること。ユーザー専用の機械語サブルーチンを定義すること。

機能 機械語サブルーチンを定義し開始アドレスを設定します。

書式 DEF　USR[番号]＝開始アドレス

使用例

100　DEF USR=0	←0番地から機械語サブルーチンを設定します。
200　A=USR(5)	←機械語サブルーチンを実行します。

解説

- 機械語サブルーチンの**開始アドレス**を定義します。
- **番号**は、複数の機械語サブルーチンを区別するもので、0から9までの数字です。したがって、同時に10個のサブルーチンを定義することができます。番号を省略した場合は0を指定したことになります。
- DEF USR文で定義した機械語プログラムはUSR関数により実行します。
- **開始アドレス**はDEF SEG文で定義したセグメントアドレスのオフセットアドレスです。USR関数を実行する前にDEF SEG文によりセグメントアドレスを定義しておく必要があります。

参照

BLOAD(機械語プログラムのロード)
CALL(機械語プログラムの実行)
DEF SEG(セグメントアドレスの設定)
USR(機械語プログラムの実行)
日本語Disk BASICリファレンスマニュアル　第10章　機械語プログラム

DELETE

【デリート】

DELETE
削る、削除するの意味。プログラムを削除すること。

D

機　能　行番号を指定して、プログラムを部分的に削除します。

書　式　DELETE　[始めの行番号] [ー終わりの行番号]

使用例　DELETE 40ー200　←40行から200行を削除します。

解　説

- **始めの行番号**から、**終わりの行番号**までを削除します。
- **始めの行番号**と**終わりの行番号**は、それぞれ省略することができます。省略した場合の、削除する範囲は次のとおりです。

書　式	削除の範囲
始めの行番号ー終わりの行番号	始めの行番号から終わりの行番号まで
始めの行番号	指定した行のみ削除
ー終わりの行番号	プログラムの最初から終わりの行番号まで

- ピリオド(.)は、直前に参照した行の行番号の代りとして使用できます。
- 指定した**終わりの行番号**を持つ行が存在しないときには、Undefined line number(行番号が存在しない)エラーになります。指定した**始めの行番号**を持つ行が存在しないときはそれより大きい最初の行番号を指定したことになります。

参　照　NEW(プログラム全体の削除)

DIM

【ディメンション】

DIMension
次元、容積の意味が転じて、配列の宣言を行うこと。

機能　配列の定義を行います。

書式　DIM　変数名(添字の最大値 [, 添字の最大値] ・・・)

使用例

```
DIM L%(100)         ←整数型配列を定義します。
DIM B$(50)          ←文字型配列を定義します。
DIM A(100, 100)     ←2次元配列を定義します。
```

解説

- **変数名**で定義した変数を配列変数として定義します。
- (　)内の**添字の最大値**の個数は、配列の次元を示します。2次元以上の配列では、各添字をカンマ(,)で区切って指定します。
- **添字の最大値**は、指定できる添字の大きさを示します。添字の最小値はOPTION BASE文により、0か1に指定することができます。指定した最小値や最大値を超える値を指定するとSubscript out of range(範囲外の添字)エラーになります。
- DIM文で宣言しなくても配列変数として使用することができます。ただし、この場合の添字の最大値は10になります。
- 配列の次元は最大255、変数の型による**添字の最大値**は次のとおりです。

型	最大値
整数型および文字型	16382
単精度実数型	16382
倍精度実数型	8190

ただし、これらの値は配列領域の大きさや、文字列の長さなどにより制限を受けます。

- 配列変数の削除にはERASE文を用います。

参照

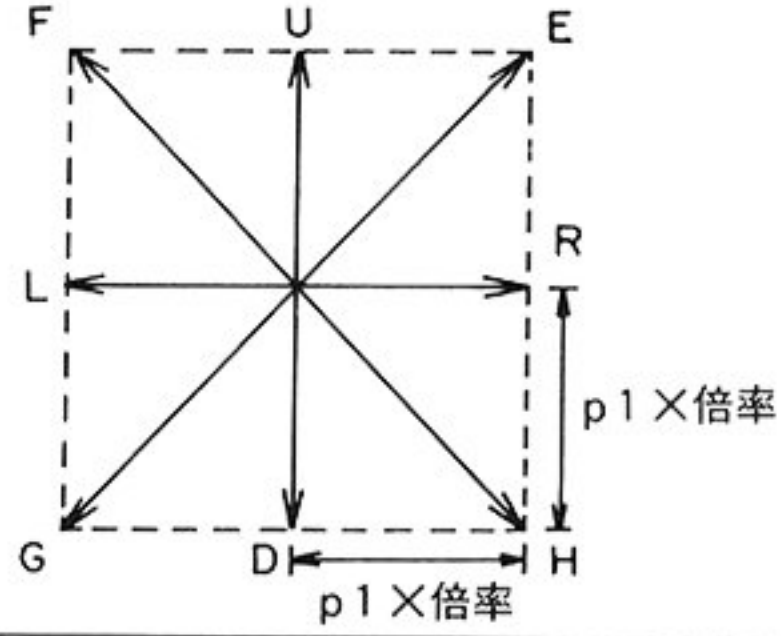
ERASE(配列変数の削除)

OPTION BASE(添字の最小値の設定)

DRAW

【ドロウ】

DRAW
線を引いて図形などを描くこと。

機能 作図命令で構成した文字列に従って、図形を描きます。

書式 DRAW ”作図命令文字列”

使用例 DRAW "R50D50L50U50" ←右に50、下に50、左に50、上に50の順にペンを移動し四角形を描きます。

解説

- **作図命令文字列**で指定した図形を画面に描きます。
- **作図命令文字列**は以下の**作図命令**によって構成する255文字以内の文字列です。
- **作図命令**には**移動命令、移動条件**を設定するもの、**変数**を指定するもの、**色**を指定するものがあります。
- **作図命令**は、アルファベット1文字か、さらにそれに対応する1または2つのパラメータで構成します。
- パラメータはコマンドに対応する数値定数または数値変数です。数値定数に指数形式を使用することはできません。また数値定数を16進数表記する場合は、後ろに必ずセミコロン(;)を付けてください。

	作図命令	命令の機能
座標指定	Ap1 (Axis)	作図命令の座標系を設定します。p1にはそれぞれ以下の座標系を0から3までの値で指定します。 0: X Y 1: X Y 2: Y X 3: Y X この命令によって指定した座標系は、DRAW 文の作図命令にのみ有効で、ワールド座標は変化しません。初期設定は0です。
移動条件	B (Blank)	移動命令の前にBを指定すると、線を引かずに(ペンを持ち上げた状態)移動します。
	N (No transition)	移動命令の前にNを指定すると、移動命令の位置まで線を引き、その後、(0,0)の位置まで戻します。
移動命令	Dp1 (Down)	最終座標から現座標系のY軸の正の方向へ直線を引きます。直線の長さはp1×倍率です。
	Ep1	最終座標から現座標系のX軸の正の方向に沿ってp1×倍率分、Y軸の負の方向に沿ってp1×倍率分移動した点を結ぶ直線を引きます。

	作図命令	命令の機能
移動命令	Fp1	最終座標から現座標系のX軸の負の方向に沿ってp1×倍率分、Y軸の負の方向に沿ってp1×倍率分移動した点を結ぶ直線を引きます。
	Gp1	最終座標から現座標系のX軸の負の方向に沿ってp1×倍率分、Y軸の正の方向に沿ってp1×倍率分移動した点を結ぶ直線を引きます。
	Hp1	最終座標から現座標系のX軸の正の方向に沿ってp1×倍率分、Y軸の正の方向に沿ってp1×倍率分移動した点を結ぶ直線を引きます。
	Lp1 (Left)	最終座標から現座標系のX軸の負の方向へ直線を引きます。直線の長さはp1×倍率です。
	Rp1 (Right)	最終座標から現座標系のX軸の正の方向へ直線を引きます。直線の長さはp1×倍率です。
	Up1 (Up)	最終座標から現座標系のY軸の負の方向へ直線を引きます。直線の長さはp1×倍率です。
	座標系の指定が0の場合の例： F U E R L p1×倍率 G D H p1×倍率	
	Mp1, p2 (Move)	最終座標からワールド座標系の(p1, p2)まで直線を引きます。
	Qp1, p2	最終座標から現座標系のX軸の正の方向に沿ってp1×倍率分、Y軸の正の方向に沿ってp1×倍率分移動した点を結ぶ直線を対角線とする長方形を描きます。
	Wp1, p2	最終座標から現座標系のX軸の正の方向に沿ってp1×倍率分、Y軸の正の方向に沿ってp2×倍率分移動した点を結ぶ直線を引きます。
	Sp1 (Scale)	移動命令の距離を指定した倍率p1で拡大または縮小して描きます。p1は0より大きい値でなければなりません。初期値は1です。
	P [p1] (Paint)	最終座標を含み、p1のパレット番号で描かれている線で囲まれた領域を塗りつぶします。p1を省略すると、Cコマンドで指定したパレット番号を指定したことになります。
	Tp1, p2	Qコマンドと同じように長方形を描き、さらに、その内部をZコマンドで指定した値で塗りつぶします。
	Yp1	D, E, F, G, H, L, M, Q, R, U, Wコマンドで描画する直線または長方形のラインスタイルを設定します。p1は&HOから&HFFFFの値でラインスタイルを設定します。初期値は&HFFFFです。ラインスタイルについてはLINE文を参照してください。
色の指定	Cp1 (Color)	パレット番号で作図命令で描く図形の色を指定します。初期値は、COLOR文で設定する前景色です。

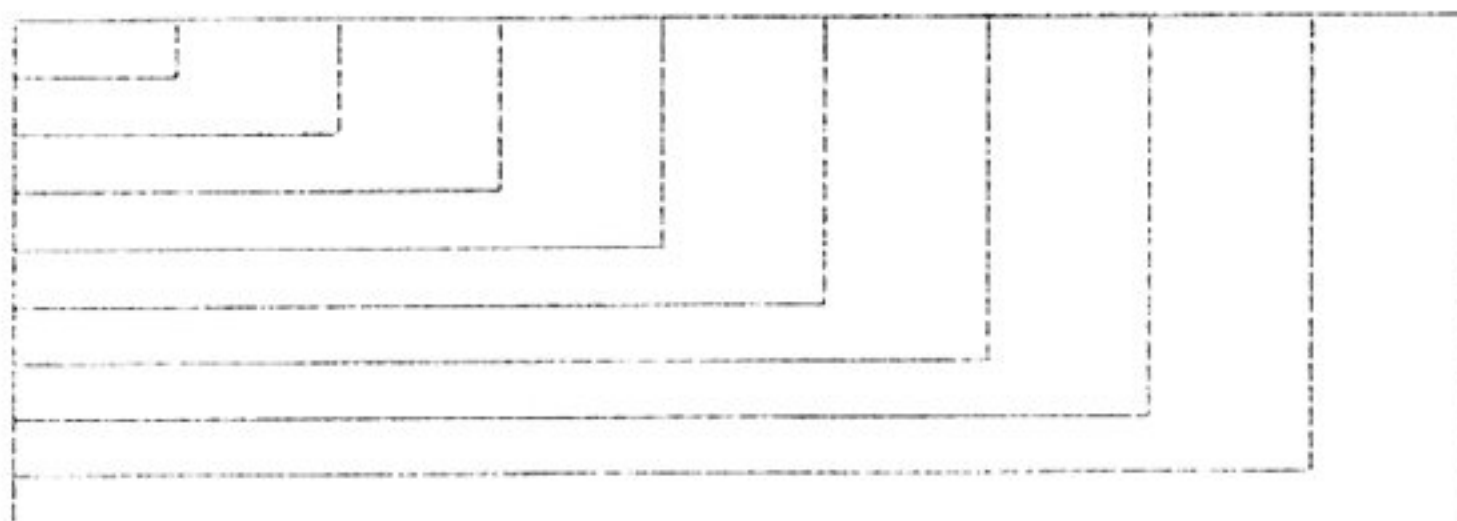
	作図命令	命令の機能
色の指定	Zp1	TおよびPコマンドで塗りつぶす色、またはタイルストリングの模様を設定します。p1は、パレット番号またはタイルストリングを表す文字変数です。初期値はCコマンドで設定するパレット番号です。タイルストリングについてはPAINT文を参照してください。
変数指定	X＝文字変数名；	作図命令文字列の一部を、文字変数で指定することができます。指定する文字変数には、DRAW文実行前に、作図命令に相当する文字列を与えておく必要があります。
	＝数値変数名；	作図命令のパラメータp1、p2などを変数として与えます。DRAW文実行前に数値変数に数値を与えておけば、作図命令はその値で実行します。 • D, E, F, G, H, L, M, Q, T, U, Wコマンドを実行すると最終参照座標は、実行後の位置に移動します。ただし、Nコマンドを実行している場合は(0, 0)に移動します。 • A, C, S, Y, Zコマンドの指定は、新たに設定し直すまで有効です。これを初期値に戻すにはSCREEN文を実行します。

参　照

LINE(線や四角形を描く)

PAINT(指定した領域を塗りつぶす)

プログラム例

```
100 ' DRAW
110 SCREEN ,,,1 : CLS  3
120 A$="R50D20L50U20"
130 FOR I=1 TO 9
140   DRAW "S"+STR$(I)+"X=A$;"
150 NEXT
160 LOCATE 0,23 : PRINT "何かキーを押すと終了します。"
170 IF INKEY$="" THEN 110
180 END
```

DSKF

【ディスク・ファンクション】

DISK Function
ディスク(disk)に関する情報を与える関数(function)。

機能　ディスクに関する種々の情報を与えます。

書式　DSKF(デバイス名 [, 機能])

使用例　PRINT DSKF(1)　←ドライブ1のディスクの残りの領域の大きさを調べます。

解説
- **ドライブ番号**で指定したディスクの種々の情報を与えます。
- **機能**は情報の種類を区別するもので0から10までの数値を指定します。

機能	内容
省略	ディスクの使用可能な残りの領域の大きさをクラスタ単位で与えます。
0	最大トラック番号(片面当たりのトラック数－1)
1	1トラック当たりのセクタ数
2	最大サーフェイス番号(ディスクのサーフェイス数－1)
3	1トラック当たりのクラスタ数 40MBのハードディスクでは1クラスタ当りのトラック数
4	ディスク当たりのクラスタ数
5	ディレクトリのあるトラック番号
6	1クラスタ当たりのセクタ数
7	FATの開始セクタ番号
8	FATの終了セクタ番号
9	FATの数
10	ディスク属性の入っているセクタ番号(ディスクID)

参照　日本語Disk BASICユーザーズマニュアル　第13章　ディスクのファイル管理

DSKI$

【ディスク・アイ・ダラー】

DiSK Input
ディスク(disk)から直接、データを読みだす(input)。

機能　ディスク上のセクタを直接指定して、データを読み出します。

書式　DSKI$(ドライブ番号, [サーフェイス番号,] トラック番号, セクタ番号)

使用例　D$=DSKI$(1 , 0 ,19, 1)　←ドライブ1、サーフェイス0、トラック19、セクタ1のデータをディスクから直接読み出します。

解説
- **ドライブ番号**, **サーフェイス番号**, **トラック番号**, **セクタ番号**で指定したセクタのデータをシステムバッファに読み出します。ディスクのセクタの大きさは256バイトであり、256バイトのデータを読み出します。
- **サーフェイス番号**, **トラック番号**, **セクタ番号**で指定できる値の範囲は、**ドライブ番号**で指定したディスクドライブの種類によって異なります。範囲外の値を指定すると bad track/sector (指定したトラック、セクタがない)エラーが発生します。指定できる範囲については日本語 Disk BASIC ユーザーズマニュアル「第13章　ディスクのファイル管理」を参照してください。
- システムバッファに読み出したデータは、文字変数に代入します。しかし、文字列として扱える長さは255バイトであるため、最後の1バイトのデータを読みだすことができません。したがって、次のような方法でデータを読み出すようにします。

1. ランダムファイルを読み出すのと同じように、FIELD 文によって、システムバッファへ複数の文字列を割り当てる方法

```
100 FIELD  #0, 128  AS  A$, 128  AS  B$
110 D$=DSKI$(D, H, T, S)
```

A$に前半の128バイトを、B$に後半の128バイトを代入します。ただし D$ にも先頭から255バイト分のデータを代入します。

2. VARPTR 関数を用いて、システムバッファの位置を直接調べ、PEEK 関数などで、メモリから直接データを読みだす方法

```
100 SEGADR=VARPTR(# 0,1):OFFSET=VARPTR(# 0,0)
110 DEF SEG=SEGADR
120 BUFADR=PEEK(OFFSET+32)+256*PEEK(OFFSET+33)
130 D$=DSKI$(D,H,T,S)
140 FOR I=0 TO 255
150         PRINT  HEX$(PEEK(BUFADR+I));
160 NEXT
```

参照
DSKO$(セクタ指定の直接書き込み)
FIELD(バッファの割当)
DSKF(ディスクに関する情報を調べる)
VARPTR(変数などアドレスを調べる)

プログラム例

```
100 ' DSKF / DSKI$
110 ' --- ディレクトリのダンプ  ---
120 DIM D$(255)
130 INPUT "Drive No.= ";D
140 T=DSKF(D,5) : MAXS=DSKF(D,1)
150 FOR I=0 TO 255
160   FIELD #0,I AS DUMMY$,1 AS D$(I)
170 NEXT
180 '
190 FOR S=1 TO MAXS
200   PRINT "Drive :";D;"     Surface : 0    Track :";
210   PRINT T;"   Secter :";S
220   DUMMY$=DSKI$(D,0,T,S)
230   FOR I=0 TO 255  STEP 16
240   ASCDAT$=""
250     FOR J=0 TO 15
260       C=I+J
270       PRINT RIGHT$("0"+HEX$(ASC(D$(C))),2);" ";
280       IF D$(C)<" " THEN LSET D$(C)="."
290       ASCDAT$=ASCDAT$+D$(C)
300     NEXT : PRINT " ";ASCDAT$
310   NEXT : PRINT : IF INKEY$<>"" THEN  330
320 NEXT
330 END

RUN
Drive : 1       Surface : 0    Track : 35    Secter : 1
65 67 64 69 63 6D 64 69 63 00 47 FF FF FF FF FF  egdicmdic.G
65 67 64 69 63 75 64 69 63 00 45 FF FF FF FF FF  egdicudic.E
65 67 7A 69 70 6D 64 69 63 00 41 FF FF FF FF FF  egzipmdic.A
65 67 7A 69 70 75 64 69 63 00 38 FF FF FF FF FF  egzipudic.8
42 4D 45 4E 55 20 20 20 20 80 37 FF FF FF FF FF  BMENU    _7
62 6D 65 6E 75 20 20 20 20 80 33 FF FF FF FF FF  bmenu    _3
42 4D 45 4E 55 20 41 55 54 80 32 FF FF FF FF FF  BMENU AUT_2
42 4D 45 4E 55 20 42 4B 50 80 30 FF FF FF FF FF  BMENU BKP_0
62 6D 65 6E 75 20 62 6B 70 01 2F FF FF FF FF FF  bmenu bkp./
42 4D 45 4E 55 20 43 50 59 80 2E FF FF FF FF FF  BMENU CPY_.
62 6D 65 6E 75 20 63 70 79 01 2C FF FF FF FF FF  bmenu cpy.,
42 4D 45 4E 55 20 44 43 4D 80 2B FF FF FF FF FF  BMENU DCM_+
42 4D 45 4E 55 20 44 43 31 80 29 FF FF FF FF FF  BMENU DC1_)
42 4D 45 4E 55 20 44 43 32 80 28 FF FF FF FF FF  BMENU DC2_(
62 6D 65 6E 75 20 64 63 64 01 27 FF FF FF FF FF  bmenu dcd.'
62 6D 65 6E 75 20 64 63 6A 01 1F FF FF FF FF FF  bmenu dcj..

Drive : 1       Surface : 0    Track : 35    Secter : 2
62 6D 65 6E 75 20 64 63 72 01 19 FF FF FF FF FF  bmenu dcr..
62 6D 65 6E 75 20 65 6E 74 01 7E FF FF FF FF FF  bmenu ent.~
42 4D 45 4E 55 20 46 4D 54 80 7F FF FF FF FF FF  BMENU FMT_
62 6D 65 6E 75 20 66 6D 74 01 81 FF FF FF FF FF  bmenu fmt._
42 4D 45 4E 55 20 48 44 4D 80 82 FF FF FF FF FF  BMENU HDM_▬
62 6D 65 6E 75 20 68 64 6D 01 84 FF FF FF FF FF  bmenu hdm.■
42 4D 45 4E 55 20 4D 4B 46 80 85 FF FF FF FF FF  BMENU MKF_■
                       :
                       :
```

DSKO$

【ディスク・オー・ダラー】

DiSK Output
ディスク(disk)に直接データを書き込む(output)。

機能　ディスク上のセクタを直接指定して、データを書き込みます。

書式　DSKO$ ドライブ番号, [サーフェイス番号,] トラック番号, セクタ番号

使用例　DSKO$ 1, 1, 19, 1　←ドライブ1、サーフェイス1、トラック19、セクタ1に直接データを書き込みます。

解説

- **ドライブ番号, サーフェイス番号, トラック番号, セクタ番号**で指定したセクタにシステムバッファのデータ256バイトを書き込みます。
- **サーフェイス番号, トラック番号, セクタ番号**で指定できる値の範囲は、**ドライブ番号**で指定したディスクドライブの種類によって異なります。範囲外の値を指定すると bad track/sector (指定したトラック、セクタがない)エラーが発生します。指定できる範囲については日本語 Disk BASIC ユーザーズマニュアル「第13章　ディスクのファイル管理」を参照してください。
- システムバッファには、次のように書き込むデータを準備します。

1. ランダムファイルを読み出すのと同じように、FIELD 文によって、システムバッファへ複数の文字列を割り当てる方法

```
100 FIELD  #0, 128  AS  A$, 128  AS  B$
110 LSET  A$=D1$
120 LSET  B$=D2$
130 DSKO$  D, H, T, S
```

A$に前半の128バイトを、B$に後半の128バイトを設定します。

2. VARPTR 関数を用いて、システムバッファの位置を直接調べ、POKE 文で、メモリへ直接データを書き込む方法

```
100 SEGADR=VARPTR(#0, 1) : OFFSET=VARPTR(#0, 0)
110 DEF SEG=SEGADR
120 BUFADR=PEEK(OFFSET+32)+256*PEEK(OFFSET+33)
130 FOR I=0 TO 255
140    POKE BUFADR+I, D(I)
150 NEXT
160 DSKO$  D, H, T, S
```

参照

DSKI$(セクタ指定の直接読み出し)
FIELD(バッファの割当)
DSKF(ディスクに関する情報を調べる)
VARPTR(変数などアドレスを調べる)

EDIT

【エディット】

EDIT
編集する意味。プログラムの編集、修正をすること。

機　能　プログラム中の一行を表示し修正します。

書　式　EDIT　[行番号]

使用例

EDIT 10　　←10行を画面に表示します。
EDIT .　　←直前に、参照した行を表示します。

解　説

- テキスト画面を消去し、最上行に**行番号**で指定した行を表示します。カーソルは行の先頭に移動します。
- **行番号**を省略すると、プログラムの先頭の行を指定したことになります。
- **行番号**に、ピリオド(.)を指定すると、直前に参照した行(修正したり、入力した行)を指定したことになります。
- プロテクトセーブしたプログラムに対して、EDIT コマンドを実行しても Illegal function call(違法機能呼び出し)エラーとなり、修正することはできません。
- LIST ［行番号］ コマンドも行の表示を行いますが、画面の消去は行いませんし、カーソルは行の先頭に移動しません。

参　照

DELETE(プログラム行の削除)
LIST(プログラムの表示)

END

【エンド】

END
終わりの意味。プログラムの実行を終了すること。

機能　プログラムの実行を終了して、BASICのコマンドレベルに戻ります。

書式　END

使用例　IF A$="E" THEN END　　←A$が"E"ならプログラムを終了します。

解説
- END文は、すべてのファイルをクローズして、プログラムの実行を終了します。
- プログラム中にEND文がなくてもプログラムの最終行に達すればプログラムの実行は終了します。ただし、この場合はファイルのクローズは行いません。
- STOP文でもプログラムの実行を停止することができます。END文とSTOP文では次の点で異なります。

	END	STOP
実行の中断を表すメッセージ	表示しない	表示する(Break in　行番号)
オープンしているファイル	クローズする	クローズしない

参照　STOP(プログラムの実行の停止)

プログラム例

```
100 ' END
110 '----  条件 (I=30) を満たすと終了する  ----
120 CLS
130 I=1
140 PRINT "*";
150 IF I=30 THEN END
160 I=I+1
170 GOTO 140

RUN
******************************
OK
```

EOF

**【イー・オー・エフ/
エンド・オブ・ファイル】**

End Of File
ファイルの終わり。すなわち、終わりを識別すること。

機能　ファイルの終わりを調べます。

書式　EOF([#]ファイル番号)

使用例　IF EOF(1) THEN CLOSE　←ファイルの終わりに達したならばクローズします。

解説
- **ファイル番号**で指定したファイルの終わりを調べます。ファイルが終わりであれば－1(真)を、そうでなければ0(偽)を返します。
- EOF 関数は、ディスク上のシーケンシャルファイル、または通信回線に対してのみ意味を持ちます。
- シーケンシャルファイルでは、ファイルの終わりに達して次に読み込むべきデータがなくなったときに－1(真)、そうでない場合に0(偽)になります。
- 通信回線では、通信バッファに受信データがない場合に－1(真)に、そうでない場合に0(偽)になります。

参照　LOF(ファイルの大きさ)
LOC(次に読み込むデータの位置)

ERASE

【イレーズ】
ERASE
消す、削除することから、配列を消去すること。

機 能　配列変数に割り当てられていたメモリ領域を開放します。

書 式　ERASE 配列変数名 [, 配列変数名] ・・・

使用例

```
100 DIM A(100)       ←DIM 文で配列変数Aを宣言します。
200 ERASE A          ←一度定義した配列を消去します。
300 DIM A(200)       ←新しく定義することができます。
```

解 説

- 配列変数名で指定した配列変数を消去します。(指定した配列変数用のメモリ領域を別の配列変数が使用できるように開放する)
- ERASE 文によって配列変数を消去すると、その後、DIM 文と同じ名前の配列を新たに定義することができます。ERASE 文で消去せずに、同じ名前の配列変数を定義すると、Duplicate Definition(二重定義)エラーになります。
- 配列変数だけでなく、すべての変数を消去するにはCLEAR 文を使います。

参 照　CLEAR(変数の消去)
DIM(配列変数の宣言)

ERR ERL

【エラー・コード】
【エラー・ライン】

ERRor code
エラーコードの意味。ここではそのコードを保持すること。

ERror Line
エラー行の意味。ここではその行番号を保持すること。

機能 エラーの発生した行の行番号、およびそのエラーコードを保持します。

書式

```
ERR
ERL
```

使用例

IF ERR=53 AND ERL=1000　THEN　2000←エラーの発生した行が1000行で、かつ File not found（ファイルが存在しない）エラーであれば、2000行からの処理を行います。

解説

- エラーが発生すると、**ERR** 関数に、エラーの種類を示すエラーコードを設定します。**ERL** 関数には、エラーの発生した行の行番号を設定します。
- エラーコードは、それぞれのエラーに付けた番号です。エラーとエラーコードの対応は「エラーメッセージ」を参照してください。
- ERR, ERL 関数は ON ERROR GOTO 文でエラー処理の制御を行うときに使用します。
- IF THEN 文の中で ERL 変数と行番号を比較する場合、次の２点に注意してください。
- **IF ERL=100　THEN** のように ERL を等号(=)の左辺にした場合、RENUM コマンドで行番号を変更すると、右辺の行番号100も変更します。(もとの100行が変更された行番号になる)
- **IF　100=ERL THEN** のように ERL を等号(=)の右辺にすると、RENUM コマンドで行番号の変更を行っても、左辺の行番号100は変更しません。

参照

ERROR（エラーの発生）
ON ERROR GOTO（エラー処理ルーチンの設定）
第２章　エラーメッセージ

ERROR

【エラー】

ERROR
エラー(誤り)。ここではエラーを発生すること。

機　能　エラーを発生します。

書　式　ERROR　エラーコード

使用例　IF A>100 THEN ERROR 6　←A＞100のときにOverflow(桁あふれ)エラーを発生させます。

解　説
- **エラーコード**に対応するエラーメッセージを表示して、プログラムの実行を終了します。
- ただし、ON ERROR GOTO 文によるエラー処理ルーチンが用意されている場合は、通常のエラー発生と同じように、そのルーチンを実行します。
- エラーコードは0から255までの数値です。エラーコードとエラーメッセーの対応は「第2章　エラーメッセージ」を参照してください。

参　照
ERR(エラーコード)
ERL(エラー発生行)
ON ERROR GOTO(エラー処理ルーチン)
第2章　エラーメッセージ

【イクスポネンシャル】

EXPonential
指数の意味。指数関数を計算すること。

機　能　指数関数e^x（自然対数の底eのx乗）を計算します。

書　式　EXP(数式)

使用例

```
A=EXP(1)
```

←eの１乗、すなわち自然対数の底eの値を求めます。

解　説

- eの(　)内の**数式**のべき乗を計算します。(e≒2.718281828459045)
- **数式**が倍精度実数のときは、結果も倍精度で計算した値になります。他の場合は、単精度で計算した値になります。

FIELD

【フィールド】

FIELD
区画した範囲の意味。

機能　ランダムファイル用のファイルバッファを、文字変数で分割します。

書式　FIELD [#] ファイル番号, 文字数 AS 文字変数 [, 文字数 AS 文字変数・・・]

使用例

```
100 OPEN "TEST" AS #1
110 FIELD #1, 128 AS A$, 128 AS B$
```

←"TEST"というファイルをオープンして、ファイルバッファをA$, B$で分割します。

解説

- **ファイル番号**で指定したファイルのファイルバッファに、プログラム中で使用する変数名を定義します。プログラム中ではこの変数を介して、ランダムファイル間とのデータの書き込み、読み込みを行います。
- **文字数**には、**文字変数**に割り当てるバッファの大きさを、バイト数で指定します。ファイルバッファの大きさは256バイトのため、文字数の長さの合計は256バイト以下でなければなりません。これを超えると、FIELD overflow(フィールドの桁あふれ)エラーになります。
- ランダムファイルとプログラムの間でのデータの入出力にはGET#, PUT#文を使用します。また**文字変数**で指定した変数に値を代入するにはLSET, RSET 文を使用します。
- 同じファイルバッファに、複数のFIELD文を実行することができます。各FIELD文は、ファイルバッファの先頭から変数への割り当てを行い、すべての割り当てが同時に有効です。したがって、同じファイルから形式の異なるデータを読み込む場合も、あらかじめ、その形式に合う変数を割り当てておくことにより処理することができます。

参照

GET#(データの読み込み)
LSET(ファイルバッファへのデータの設定)
OPEN(ファイルのオープン)
PUT#(データの書き込み)
RSET(ファイルバッファへのデータの設定)

FILES
LFILES

【ファイルズ】
【エル・ファイルズ】

FILES
整理した書類の意味から、プログラムやデータを記録したものの集まり。ここではそれを画面に表示すること。

List FILES
表を作るという意味から転じて、ここではプリンタに出力すること。

機能

ディスク中のディレクトリ情報(ファイル名、ファイルの種類およびファイルの大きさ)を画面上あるいはプリンタに出力します。

書式

FILES　[ドライブ番号]
LFILES　[ドライブ番号]

使用例

FILES 1　←ドライブ1のすべてのファイル名を画面に表示します。

LFILES　←ドライブ1のすべてのファイル名をプリンタに出力します。

解説

• FILES文は、ドライブ番号で指定したディスク上に登録しているすべてのファイルのディレクトリ情報を、次の形式で表示します。

ファイル名：最大6文字。作成するときに付けたプログラムやファイルの名前
ファイル名拡張子：最大3文字。一般にファイルの種類をユーザーが区別するのに使用します。
ファイルの種類：ファイルの種類を示す1文字の記号

記号	ファイルの種類
空白(" ")	アスキーセーブしたプログラムファイル、またはOPEN文で作成したデータファイル
ピリオド(.)	バイナリセーブしたプログラムファイル
アスタリスク(*)	BSAVE文で作成した機械語ファイル

ファイルの大きさ：クラスタ単位

• **ドライブ番号**を省略すると1を指定したことになります。
• LFILES文は、ディレクトリ情報をプリンタに出力します。

参照

BSAVE(機械語プログラムのセーブ)
OPEN(データファイルの作成)
SAVE(プログラムのセーブ)

FIX

【フィックス】

FIX
直す、整理するという意味から、ここでは数値を整数にすること。

機　能　　数値の小数部を切り捨てて整数にします。

書　式　　FIX(数式)

使用例　　A=FIX(78.56)　　←78.56の小数部を切り捨てます。

解　説

- ( )内の**数式**の値の小数部を切り捨てます。
- FIX関数は、数値が正の数でも負の数でも、単純に小数部を切り捨てて整数にします。一方、同じように小数部を切り捨てるINT関数は、数値を超えない最大の整数を求めるため、数値が負の場合には結果が異なります。

```
FIX( 1.28)──> 1     INT( 1.28)──> 1
FIX(−1.28)──>−1     INT(−1.28)──>−2
```

- CINT関数とは異なり型の変換は行いません。

参　照　　CINT(小数部の四捨五入)
INT(小数部の切り捨て)

FOR〜TO〜NEXT

【フォー・トゥ・ネクスト】

FOR〜TO〜NEXT
〜から〜までの間を〜まで繰り返す、といった意味。したがって、プログラムを繰り返し実行すること。

機能 FOR文からNEXT文の間にある一連の命令を、特定の回数だけ繰り返し実行します。

書式

```
FOR　変数名=始めの値　TO　終わりの値 [STEP 増分]
　 :
NEXT　[変数名]
```

使用例

```
100 FOR I=1 TO 10
110     PRINT I;
120 NEXT
```

←PRINT I；をIが1から10までの間繰り返し実行します。

解説

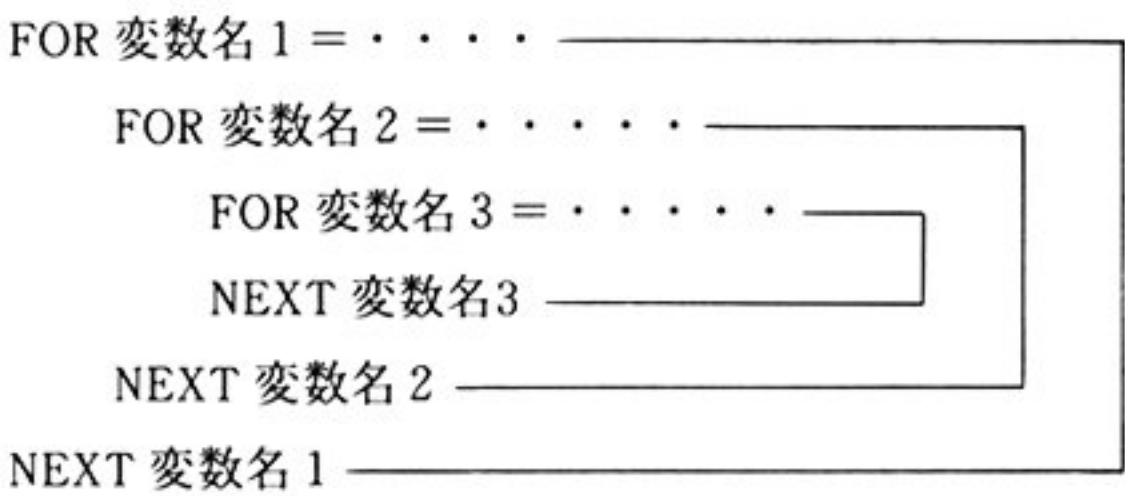
- 繰り返す一連の命令の最初にFOR文を、最後にNEXT文を置きます。この集まりをFOR〜NEXTループと言います。
- FOR文の後ろの**変数名**は、FOR〜NEXTループを何回繰り返したかを数える変数の名前です。変数には整数型、あるいは単精度実数型を指定します。
- 変数の値は、**始めの値**を初期値として、FOR〜NEXTループを繰り返すたびに、**増分**で指定した値で増減します。増減後の変数の値を、**終わりの値**と比較し、**終わりの値**よりも大きく(または、小さく)なると、プログラムの実行はNEXT文の次の文に移ります。**増分**を省略すると1を指定したことになります。
- NEXT文の後ろには、FOR文で指定したものと同じ**変数名**を置きます。NEXT文の後ろの**変数名**を省略すると、直前で実行しているFOR文の変数名に対応します。
- 繰り返しの回数は、**始めの値**、**終わりの値**、**増分**によって決定します。次の場合は、FOR〜NEXTループは一度も実行しません。

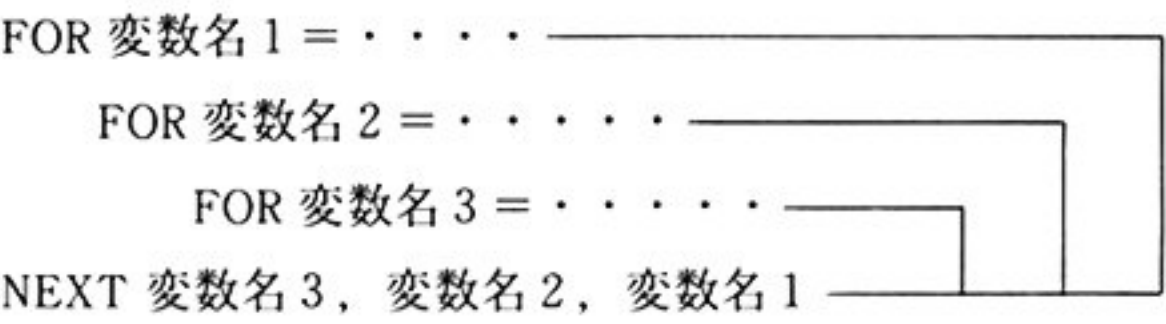
  始めの値>終わりの値 かつ 増分>0

  始めの値<終わりの値 かつ 増分<0
- **始めの値**=**終わりの値**で、かつ**増分**が0でない場合はFOR〜NEXTループを一度だけ実行します。
- **増分**が0の場合、FOR〜NEXTループを無限に繰り返します。したがって、何らかの条件によりループの外に分岐させる必要があります。
- FOR〜NEXTループは、一つのFOR〜NEXTループの中に、いくつかのFOR〜NEXTループをおいて、入れ子構造(ネスティング)を作ることができます。このとき、内側のループは完全に外側のループに含まれていなければなりません。

正しい入れ子構造	誤った入れ子構造
FOR 変数名 1 = ・・・・ 　FOR 変数名 2 = ・・・・ 　NEXT 変数名 2 NEXT 変数名 1	FOR 変数名 1 = ・・・・ 　FOR 変数名 2 = ・・・・ 　NEXT 変数名 1 NEXT 変数名 2

また、それぞれのFOR～NEXT ループには異なった変数名を使用しなければなりません。

```
FOR 変数名 1 = ・・・・
  FOR 変数名 2 = ・・・・・
    FOR 変数名 3 = ・・・・・
    NEXT 変数名3
  NEXT 変数名 2
NEXT 変数名 1
```

• 複数のFOR～NEXT ループが同一行で終わる場合には、各々のNEXT 文をカンマ(,)で区切って一つにまとめることができます。

```
FOR 変数名 1 = ・・・・
  FOR 変数名 2 = ・・・・・
    FOR 変数名 3 = ・・・・・
NEXT 変数名 3, 変数名 2, 変数名 1
```

• FOR 文と NEXT 文が対応しない場合、FOR without NEXT(NEXT 文のない FOR 文)エラー、あるいは NEXT without FOR(FOR 文のない NEXT 文)エラーになります。

参　照　WHILE～WEND(繰り返し)

FPOS

【ファイル・ポジション】

File POSition
ファイルの位置の意味。

機能

ディスク上の物理的な現在位置を知らせます。またプリンタ出力の論理的な現在位置を知らせます。

書式

FPOS([#]ファイル番号)

使用例

FP=FPOS(1)　←直前に読み書きしたセクタ位置を変数FPに代入します。

解説

- **ファイル番号**で指定したファイルがディスクファイルであれば、直前に読み書きを行ったセクタの番号を返します。このときのセクタの番号は、サーフェイス0、トラック0、セクタ1を0として通し番号で付けた値です。
- **ファイル番号**で指定したファイルがプリンタであれば、プリンタ出力の論理的位置を返します。LPOS 関数と同じ機能になります。

参照

LOC(ファイル中の現在位置)
LPOS(プリンタ出力の論理位置)

FRE

【フリー】

FREe
自由な、開放的なという意味から、空いているユーザーエリアのこと。

機　能

BASIC で使用可能な作業空間(ユーザーエリア)のうち、現在使用していない領域の大きさをバイト数で示します。

書　式

FRE(機能)

使用例

```
PRINT FRE(1)
```

←プログラムの入る領域の大きさを表示します。

解　説

• **機能**には0から3まで整数を指定します。それぞれの意味は次のとおりです。

機能	意　　味
0	変数領域(単純変数および文字列)として使用する領域の残りバイト数
1	プログラム領域として使用する領域の残りバイト数
2	変数領域とプログラム領域で使用する領域の合計バイト数
3	配列変数領域として使用する領域の残りバイト数

• **機能**が0または2のときは、領域の圧縮処理を行うため、関数値を返すまで、時間がかかることがあります。BASIC では文字列を可変長として扱っているため、文字列の長さが変わると、無駄な空間を生じることがあります。圧縮処理は、この無駄な領域を詰めて、使用可能な領域として開放することです。

参　照

CLEAR(変数の消去)

GET#

【ゲット】

GET
得る、手に入れることから転じて、データを読み取ること。

機能　ランダムファイルのデータをファイルバッファに読み込みます。

書式　GET[#]ファイル番号 [, レコード番号]

使用例　GET#1, 1　←ファイル番号1のファイルの1番目のレコードを読み込みます。

解説

- **ファイル番号**で指定したランダムファイルから、**レコード番号**で指定したレコードをファイルバッファに読み込みます。
- **レコード番号**を省略すると、直前のGET文あるいはPUT文で指定したレコード番号の次のレコード番号を指定したことになります。ランダムファイルをオープンした直後は1になります。
- **レコード番号**は1から65000までの数値を指定します。
- GET文で読み取ったデータは、FIELD文でファイルバッファに割り当てた変数から、プログラムに引き渡します。
- キーボードファイル(KYBD:)に対しても、GET文を使用することができます。このとき、レコード番号はキーボードから読み取る文字数を意味し、0から255までの数値を指定します。数値を省略したとき、または0を指定したときは256文字を読み取ることを意味します。キーボードファイルはあらかじめ、入力モードでオープンしてください。

参照

FIELD(ランダムファイルのバッファの割り当て)
OPEN(ファイルのオープン)
PUT#(ランダムファイルへの書き込み)

GET@

【ゲット・アットマーク】

GET
得る、手に入れることから転じて、データを読み取ること。

機　能　画面上の指定領域のグラフィックパターンを、数値型配列変数に読み取ります。

書　式　GET[@]（Sx1，Sy1）−（Sx2，Sy2），配列変数名[（要素番号）]

使用例

```
100  DIM A(1058)
110  GET@(100, 50)-(150, 200), A
```

←（100，50）−（150，200）を対角座標とする範囲のパターンを配列Aに読み取ります。

解　説

- スクリーン座標上の2点（**Sx1，Sx2**），（**Sx2，Sy2**）を対角座標とする四角形内の領域のグラフィックパターンを、**配列変数名**で指定した配列に読み取ります。2番目の座標は、相対座標で指定することもできます。
- **配列変数名**は、グラフィックパターンを読み取るための、数値型の配列変数の名前です。読み取ったグラフィックパターンは、指定した配列の型（整数、単精度、倍精度）に応じて分割され、配列の各要素に収められます。このため、読み取り範囲に応じた配列の大きさを、DIM文で宣言しておく必要があります。
- 必要な配列の大きさは読み取り範囲の大きさ、画面モード（カラーか白黒モード）に異なります。まず必要なバイト数を計算し、続けて配列の型に応じて必要な配列の大きさを求めます。

①　必要なバイト数

バイト数＝INT（（X＋7）/8）×Y×M＋4

X：指定範囲のX軸方向のドット数（**Sx2−Sx1**＋1）
Y：指定範囲のY軸方向のドット数（**Sy2−Sy1**＋1）
M：画面モードによる係数

白黒モード	M＝1
8色中8色モード	M＝3
4096色中8色モード	M＝3
4096色中16色モード	M＝4

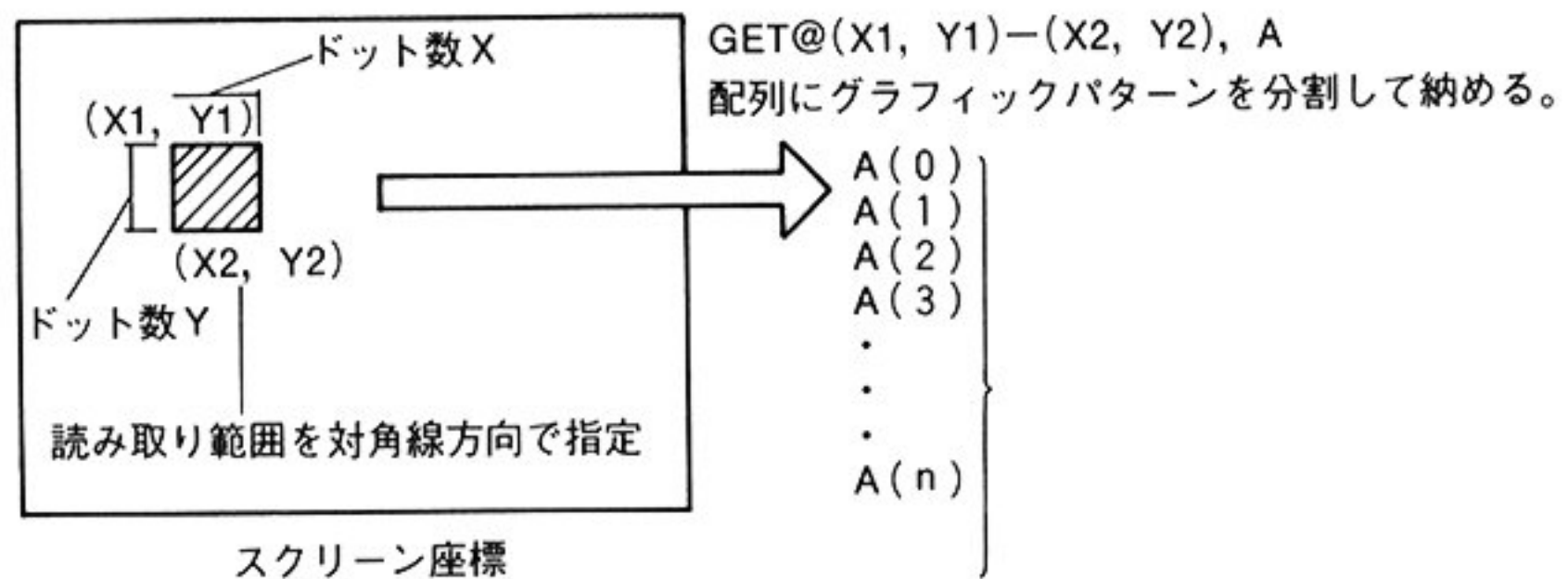
②　配列の型により必要な配列の大きさは異なります。

配列の大きさ（添字の最大値）＝必要なバイト数/配列の型＋1

配列の型：	
整数型	2
単精度型	4
倍精度型	8

（配列の大きさは、1次元配列で添字の最小値が0のときの値です）

G

読み取るグラフィックパターンに比べて、指定した配列の大きさが小さい場合は、Illegal function call(違法関数呼び出し)エラーになります。

- **要素番号**は、指定した配列のどの要素番号からデータを読み込むのかを指定するもので、省略すると配列変数の最初(0または1)になります。要素番号の指定を行うと、一つの配列に複数のグラフィックパターンを読み込むことができます。

参　照　OPTION BASE(配列の添字の最小値)
PUT @(グラフィックパターンの表示)

プログラム例

```
100 ' GET@ / PUT@
110 ' ---   図形を読み取り、複写する   ---
120 CONSOLE ,,,1 : SCREEN 0,0 : CLS 3
130 DIM A%((((51+7)¥8)*20*3+4)¥2+1)
140 '
150 LINE(0,8)-(50,27),5,BF
160 LINE(4,12)-(46,23),1,BF
170 LINE(8,16)-(42,19),6,BF
180 GET@(0,8)-(50,27),A%
190 '
200 FOR Y=50 TO 150 STEP 50
210   FOR X=50 TO 500 STEP 50
220     PUT@(X,Y),A%
230   NEXT X
240 NEXT Y
250 END
```

GOSUB～RETURN

【ゴーサブ】【リターン】

GO to SUBroutine
サブルーチンへ行くこと。プログラム中のサブルーチンを実行すること。

RETURN
戻ること。サブルーチンからもとのメインプログラムに戻ること。

G

機能　GOSUB 文でサブルーチンを実行します。サブルーチンからは RETURN 文で、GOSUB 文の次の文に戻ります。

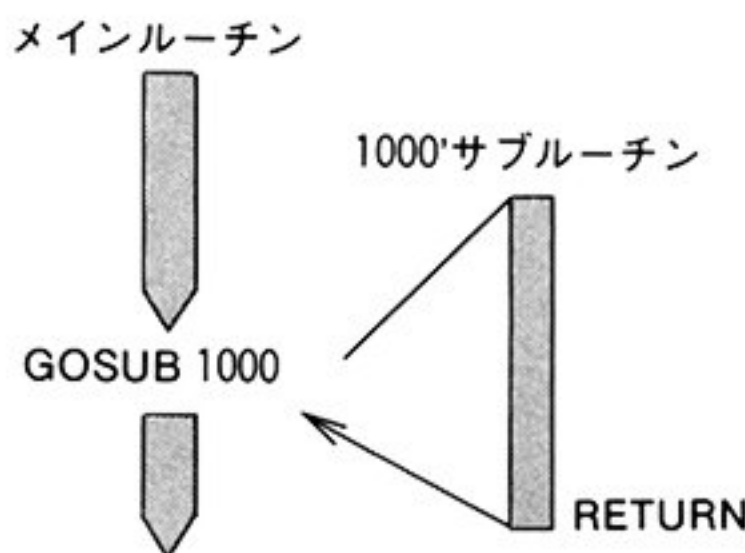
書式

```
GOSUB　行番号
RETURN　 [行番号]
```

使用例

```
100 GOSUB 1000
        ⋮
1000 'サブルーチン
        ⋮
2000 RETURN
```

解説

- GOSUB 文の後ろに指定した**行番号**(またはラベルの示す行)に制御を移します(サブルーチンの実行)。したがって、指定した**行番号**(またはラベル)がサブルーチンの開始行になります。
- GOSUB 文でサブルーチンに移り、サブルーチンプログラムを実行し、RETURN 文で、GOSUB 文の次の命令に戻ります。それ以外の場所に戻りたい場合は、RETURN 文の後ろに**行番号**を指定します。
- サブルーチンの中から、さらに別のサブルーチンを実行すること(サブルーチンの多重化)ができます。この多重化は、メモリのスタックが許すかぎり行うことができます。スタック領域が足りなくなった場合は、Out of memory(メモリが足りない)エラーになります。この場合は、CLEAR 文でスタック領域を大きく取るようにしてください。

- GOSUB 文の実行なしに、RETURN 文を実行しようとすると RETURN without GOSUB (GOSUB 文のない RETURN)エラーになります。

参照　CLEAR(スタック領域の確保)
RETURN(サブルーチンからの復帰)

GOTO

【ゴートゥ】

GO TO
go toの～へ行くという意味から、指定の行に行くこと。

機　能　指定した行へ無条件に制御を移します。

書　式

GOTO　行番号
GO TO　行番号

使用例

```
100　INPUT A, B
110　PRINT A+B
120　GOTO　100
```

←A, Bの加算結果を表示後、再び、A, Bの入力を待ちます。

解　説

・行番号(またはラベル)で指定した行に無条件に制御を移します。GOTOとGO TO(GOとTOの間に1個だけ空白)の機能の違いはありません。

参　照　ON GOTO(条件付きで制御を移す)

プログラム例

```
100 ' GOTO
110 ' ---  Eキーを押すと、終了する(無条件ジャンプ) ---
120 PRINT "Eキーを押してください。"
130 K$=INKEY$
140 IF K$="E" OR K$="e" THEN GOTO *QUIT
150 GOTO 120
160 *QUIT
170 END
```

HELP ON
HELP OFF
HELP STOP

【ヘルプ・オン】
【ヘルプ・オフ】
【ヘルプ・ストップ】

HELP key
HELPキーを意味する。これからHELPキーに関する制御を行うこと。

機能　HELP キーによる割り込み処理ルーチンの実行を許可、禁止、停止します。

書式

```
HELP |ON  |
     |OFF |
     |STOP|
```

使用例

HELP　ON	←割り込み処理ルーチンの実行を許可します。
HELP　OFF	←割り込み処理ルーチンの実行を禁止します。
HELP　STOP	←割り込み処理ルーチンの実行を停止します。

解説

- HELP キーを押したときに実行する割り込み処理ルーチンの制御を行います。
- 割り込み処理ルーチンはON HELP GOSUB文で定義します。
- **HELP ON** は割り込み処理ルーチンの実行を許可します。この命令を実行後は、HELP キーを押すとON HELP GOSUB文で定義した割り込み処理ルーチンを実行します。
- **HELP OFF** は割り込み処理ルーチンの実行を禁止します。この命令を実行後は、HELP キーを押しても割り込み処理ルーチンは実行せず、HELP キー本来の動作を行います。プログラム終了時には必ず、HELP OFFを実行してください。
- **HELP STOP** は割り込み処理ルーチンの実行を停止します。この命令を実行後は、HELP キーが押されたことを記憶し、割り込み処理ルーチンを実行しません。このあと、HELP ONを実行すると、割り込み処理ルーチンを実行します。

参照　ON HELP GOSUB(HELPキーによる割り込み処理ルーチンの定義)

H

HEX$

【ヘキサ・ダラー】

HEXadecimal
16進数を意味する hexadecimal の省略形。

機能　10進数を16進表記の文字列に変換します。

書式　HEX$(数式)

使用例　PRINT HEX$(N)　←Nに相当する10進数を16進文字列に変換し、画面に表示します。

解説
- 10進数の**数値**を16進表記の文字列に変換します。
- **数値**には−32768から65535の範囲の**値**を指定します。**数値**が負の数の場合、2の補数形式で表現します。したがって、**数値**を−Nとした場合、HEX$(−N)とHEX$(65536−N)は同じ結果になります。

例

PRINT HEX$(−1)	PRINT HEX$(65535)
FFFF	FFFF
OK	OK

- 16進表記の数値は、16進数の前に&H(または&h)を付けて表現します。
- HEX$関数で文字列に変換したものを、10進数の数値に変換するにはVAL("&H"+A$)のようにします。

参照　OCT$(10進数を8進表記の文字列に変換)
VAL(文字列を数値に変換)

プログラム例

```
100 ' HEX$ / OCT$
110 ' --- 10進数 -> 16進数 -> 8進数 ---
120 PRINT "10進数  16進数  8進数"
130 FOR I=0 TO 255
140   PRINT USING " ###";I;
150   PRINT TAB(12);RIGHT$(" "+HEX$(I),2);
160   PRINT TAB(21);RIGHT$("  "+OCT$(I),3)
170 NEXT
180 END
```

```
run
10進数  16進数  8進数
   0       0      0
   1       1      1
   2       2      2
   3       3      3
   4       4      4
   5       5      5
   6       6      6
   7       7      7
   8       8     10
   9       9     11
  10       A     12
  11       B     13
  12       C     14
  13       D     15
  14       E     16
  15       F     17
  16      10     20
  17      11     21
```

IF～THEN～ELSE
IF～GOTO～ELSE

【イフ・ゼン・エルス】

IF～THEN～ELSE
もし～ならば～、そうでなければ～、という意味から条件判断を行って、分岐すること。

【イフ・ゴウトゥ・エルス】

IF～GOTO～ELSE
もし～ならば～へ、そうでなければ～、という意味から条件判断を行って、分岐すること。

機能　条件判断を行って、次に実行すべきプログラム行を選択します。

書式

```
IF 条件式 |THEN | 文    ||        | 文    |
          |     |行番号 ||  [ELSE |行番号 | ]
          |GOTO  行番号 ||
```

使用例

IF A＝0 THEN PRINT B ELSE PRINT C　←Aが0ならばBを、そうでなければCを表示します。

IF A＝0 GOTO 100 ELSE 200　←Aが0ならば100行へ、そうでなければ200行へ分岐します。

解説

- **条件式**は、演算結果が真(－1)または偽(0)で得られる関係演算式です。
- **条件式**を実行し、真であればTHENあるいはGOTO以下の文や行を実行します。また偽であればELSEに続く文を実行します。
- ELSE以降は省略することができ、省略した場合、**条件式**の結果が偽であれば次の行を実行します。
- THEN, ELSEの後ろには1つの文、またはコロン(：)で区切って並べた一連の文を指定します。単に数値だけを指定すると、分岐先の行番号を指定したことになります。GOTOの後ろには行番号しか指定できません。
- THEN、ELSEに、別のIF文を続けて、多重構造にすることができます。このとき、THENとELSEの数が合わないと、各々のELSEは最も近くにあって、まだ対応付けしていないTHENと対を作ります。多重構造にできるのは1行に書ける範囲です。
- 条件式で単精度または倍精度の値を特定の値と比較する場合は、等号(＝)ではなく不等号(<、>)を利用するようにします。これは、単精度や倍精度の数値は、内部では浮動小数点形式で記憶しており、厳密には正しい値を持たない場合があるためです(つまり近似値を持っています)。したがって、特定の値と等しいかどうかを比較する場合は、その値との差がどれだけであるかで判断するようにしてください。

INKEY$

【インキー・ダラー】

INput KEY
キー入力。すなわち、キーを入力すること。

機能　キーボードから入力した文字を与えます。キーを押していなければ空文字列(ヌルストリング)を返します。

書式　INKEY$

使用例　100 IF A$=INKEY$: IF A$="" THEN 100　←押されたキーの文字をA$に代入します。

解説
- INKEY$関数を実行したときに、押しているキーの文字を返します。何も押していないときには空文字列("")を返します。
- 押したキーの文字は、画面に表示しません。また、INPUT$関数などと異なり、キーを押すまで、入力を待ち続けることはしません。したがって、キー入力待ちをするためには、次のようにするとよいでしょう。

例
```
100 A$=INKEY$ : IF A$="" THEN 100
```

キーを押さないと100行でループしています。
- INKEY$関数は STOP キーや CTRL + C キーの文字を返すことはできません。
- プログラムで入力待ちの命令を実行していなくても、キーボードからの入力を許しており(先行入力)、押されたキーのデータはキーボードバッファに格納しています。INKEY$関数は、このキーボードバッファの先頭の一文字を返すため、先行入力しているときは示す文字が異なることがあります。先行入力しているデータを消去するには、次のようにすると良いでしょう。

例
```
100 A$=INKEY$ : IF A$<>"" THEN 100
```

参照　INPUT$(キーボードからのデータの読み込み)

INP

【インポート】

IN Port
ポートから中へ(in)ということ。I/O ポートからデータを直接読み込むこと。

機 能　I/O ポートから直接 1 バイトのデータを読み込みます。

書 式　INP(ポート番号)

使用例　A=INP(&HE0)　←ポート &HE0 からデータを読み込みます。

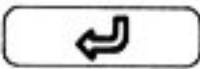
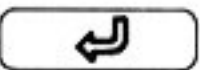
解 説
- **ポート番号**で指定した I/O ポートから 1 バイトのデータを読み込みます。
- **ポート番号**は－32768から65535までの数値を指定します。負の数を設定した場合は65536に指定した値を加算した結果を指定したことになります。

参 照　OUT(I/Oポートに直接出力)

INPUT

【インプット】

INPUT
データを入力(input)すること。

機　能　キーボードから入力したデータを変数に読み込みます。

書　式　INPUT　["入力メッセージ" {; | ,}] 変数名 [, 変数名]・・・

使用例　INPUT "ナマエヲニュウリョク" ; N$　　←"ナマエヲニュウリョク?" と表示し入力待ちになります。

解　説

- INPUT 文を実行すると **"入力メッセージ"** を表示し、キーボードからのデータ入力待ちになります。
- キーボードから**変数名**の数だけデータを入力し、［↵］キーを押すと、それぞれの変数にデータを読み込みます。
- **入力メッセージ**はデータ入力待ちの際に画面に表示する文字列で、文字定数を指定します。文字変数は使用できません。このような入力を促すメッセージをプロンプト(prompt)と呼ぶことがあります。
- 入力メッセージに**セミコロン(;)**を続けた場合は、入力メッセージの後ろに疑問符(?)、1文字分の空白とカーソルを表示します。また**カンマ(,)**を続けた場合は入力メッセージの後ろにはカーソルだけ表示します。
- **入力メッセージ**を省略した場合は疑問符(?)、1文字分の空白とカーソルを表示します。
- **複数の変数名**を指定した場合、入力するデータをカンマ(,)で区切っていきます。
- 入力したデータと変数の型は一致していなければなりません。変数の型や、入力したデータの個数が変数の数と一致していない場合は、**"?Redo from start"**(もう一度初めから入力しなさい)と表示し、データの再入力を促します。
- 何も入力せずに［↵］キーだけを押した場合は、変数が数値変数であれば0を、文字変数であれば空文字列を入力したことになります。
- 文字データの前後に入力した空白(スペース)は無視し、入力しません。入力データに空白やカンマ(,)を含む場合は、二重引用符(")で文字列を囲みます。

参　照　LINE INPUT(一行を入力)

INPUT#

【インプット・シャープ】
INPUT
データをファイルから入力(input)すること。

機能　シーケンシャルファイルからデータを読み込みます。

書式　INPUT#ファイル番号，変数［，変数］・・・

使用例　INPUT#1，A$，B$　　←データを A$，B$に読み込みます。

解説

- **ファイル番号**で指定したシーケンシャルファイルからデータを読み込みます。ディスク上のシーケンシャルファイルに限らず、通信回線やキーボードからのデータ入力に使用します。
- **変数**には、データを読み込む変数を指定します。
- 読み込むデータと変数の型は一致していなければなりません。また変数の型にしたがって、次の規則でデータを読み込みます。

(1) 変数が数値型の場合
- データの先頭の空白は無視し、これ以外の文字から空白、カンマあるいはCRコードまでのすべての文字を、数値データとして変数に読み込みます。

(2) 変数が文字型の場合
- データの先頭の空白は無視し、これ以外の文字からをデータとして読み込みます。最初の文字が二重引用符(")であれば引用符付きデータと判断し、次の二重引用符との間にある文字列を変数に読み込みます。
- 最初の文字が二重引用符でなければ引用符無しのデータと判断し、カンマあるいはCRコードまでのデータを変数に読み込みます。したがってデータにカンマやCRコードを含む場合は、データの両端に二重引用符が必要になります。ただし、いずれの場合も255文字を超える場合は、255文字までをデータとします。残りは次の変数に読み込みます。

参照　PRINT#(シーケンシャルファイルへのデータ出力)

プログラム例

```
100 ' INPUT#
110 ' ---  シーケンシャルファイルからデータを読み込む  ---
120 OPEN "2:DATA1" FOR INPUT AS #1
130 IF EOF(1) THEN GOTO 170
140   INPUT #1,NM$,TEL$
150   PRINT NM$,TEL$
160 GOTO 130
170 CLOSE
180 END
```

INPUT$

【インプット・ダラー】

INPUT String
stringは一連とか一列の意味。転じて、文字列のこと。文字列を入力(input)すること。

機 能　キーボードやディスクファイルなどから指定した長さのデータを読み取ります。

書 式　INPUT$(文字数 [, [#]ファイル番号])

使用例　A$=INPUT$(1)　　←キーボードから1文字読み込みます。

解 説
- **ファイル番号**で指定したファイルから、**文字数**で指定した文字数分のデータを読み込みます。
- **ファイル番号**を省略すると、キーボードからのデータ入力になります。ただしINPUT文のように入力したデータを画面に表示しません。
- INPUT$関数は指定した文字数分のデータを読み込むまで次の命令を実行しません。
- INPUT$関数は STOP キーや CTRL + C キーの文字を返すことはできません。

参 照　INKEY$(キーボードからのデータ入力)

プログラム例

```
100 ' INPUT$
110 ' ---  メニュー処理を作成する  ---
120 CLS
130 LOCATE 20,5 : PRINT "*****  JOB MENU  *****"
140 LOCATE 23,8 : PRINT "A : DATA APPEND"
150 LOCATE 23,10: PRINT "B : DATA DELETE"
160 LOCATE 23,12: PRINT "C : PRINT LIST"
170 LOCATE 23,14: PRINT "D : JOB END"
180 LOCATE 20,17: PRINT "**********************"
190 K$=INPUT$(1)
200 IF K$="A" THEN LOCATE 23,19:PRINT "Command : APPEND"
210 IF K$="B" THEN LOCATE 23,19 : PRINT "Command : DELETE"
220 IF K$="C" THEN LOCATE 23,19 : PRINT "Command : LIST  "
230 IF K$="D" THEN LOCATE 23,19 : PRINT "=== GOOD BY! ===" : END
240 BEEP : GOTO 190
```

INPUT WAIT

【インプット・ウェイト】

INPUT WAIT

wait は待つこと。したがって待ち時間を決めた INPUT 文。

機　能　キーボードから変数にデータを読み込みます。その際に、その入力待ち時間を制限します。

書　式　INPUT WAIT　待ち時間,［"入力メッセージ" | ; | , |］変数名［, 変数名］・・・

使用例　INPUT WAIT 300, "ナマエヲニュウリョク"; A$　←A$ にデータを読み込む際に30秒だけ待ちます。

解　説

- **待ち時間**で指定した時間だけ入力を待ちます。**待ち時間**の単位は0.1秒です。
  待ち時間には 0 を除いた－32768から65535までの整数を指定します。ただし負の数を指定した場合は65536に指定した値を加算した結果を指定したことになります。例えば **INPUT WAIT －1, A$** と **INPUT WAIT 65535, A$** は同じ時間だけ待ちます。
- この文の後ろにマルチステートメントとして他の文がある場合、待ち時間内に入力を行ったときのみ後の文を実行します。入力を行わなかったときは、後の文を実行せずに、次の行を実行しますので注意してください。
- **待ち時間**の制限があることを除けば、INPUT 文と同じ機能です。

参　照　INPUT（キーボードからのデータ入力）

I

INSTR

【インストリング】

IN STRing
string は一連とか一列の意味。転じて、文字列のこと。in は～のなかの、～についての意味の接頭語。

機　能　文字列の中から指定した文字列を探し出し、見つかればその文字列の開始位置を知らせます。

書　式　INSTR([開始位置,] 探索される文字列, 見つける文字列)

使用例　A=INSTR(10, B$, C$)　　←B$ の10文字目から C$ を探索します。

解　説

- **探索される文字列**の中を、**開始位置**から**見つける文字列**を探します。見つかれば、その位置を**探索される文字列**の先頭からのバイト数で知らせます。もし見つからない場合は、0になります。
- **開始位置**を**探索される文字列**の先頭からのバイト数で指定します。省略すると文字列の最初から探します。
- **見つける文字列**に空文字列("")を指定すると、**開始位置**と同じ値を返します。

プログラム例

```
100 ' INSTR
110 ' ---  文字列の位置を調べる  ---
120 A$="EPSON PERSONAL COMPUTER WORLD"
130 P=INSTR(A$,"COM")
140 PRINT P
150 END

RUN
 16
OK
```

INT

【イント/インテジャ】

INTeger
整数のこと。ここでは、数値を整数にすること。

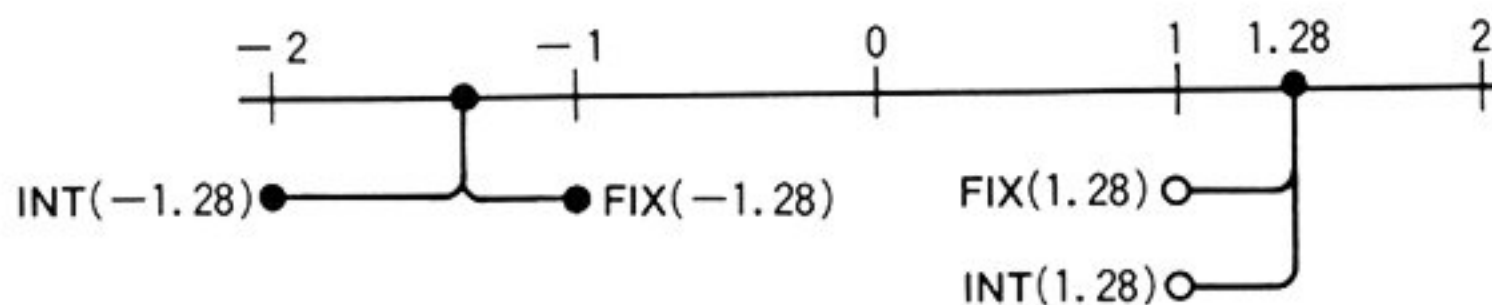
機能　数値の小数部を切り捨てて整数にします。

書式　INT(数式)

使用例　A=INT(30.65)　←30.65の小数部を切り捨てて30にします。

解説
- ( )内の**数式**の値の小数部を負の方向に切り捨てます。
- INT 関数は小数点以下を切り捨てる際に、**数値**を超えない最大の整数を求めます。したがって、単純に小数部を切り捨てる FIX 関数とは、数値が負の場合に結果が異なります。

例

FIX(　1.28)──>　1　　INT(　1.28)──>　1
FIX(−1.28)──>−1　　INT(−1.28)──>−2

- CINT 関数とは異なり、型の変換は行いません。

参照　CINT(小数部の四捨五入)
FIX(小数部の切り捨て)

JIS$

【ジス・ダラー】

JIS
Japanese Industrial Standard(日本工業規格)の意味。転じて、漢字の文字コードを求めること。

機　能　文字列の最初の２バイトを16進数形式の文字列で表します。

書　式　JIS$(文字列)

使用例　A$=JIS$(KMID$("日本", 2, 1))　←"日"のJISコード467Cを求めます。

解　説

- **文字列**の最初の２バイトを16進数形式の文字列で表します。
- 漢字などの２バイト文字の文字コードを簡単に調べることができます。ただし、この関数は単純に**文字列**の最初の２バイトの文字コードを返すだけです。

  例：A$=JIS$("日本")　これではKIコードのコードが返ります。
  　　A$=JIS$(KMID$("日本", 2, 1))　"日"のJISコードが返ります。
- 逆にJISコードから２バイト文字を得るにはKNJ$関数を用います。

J

参　照　ASC(１バイト文字の文字コードを与える)
KNJ$(２バイト文字に変更)

プログラム例

```
100 ' JIS$
110 ' ---   文字の漢字コードを調べる   ---
120 CLS
130 INPUT  "文字を入力";K$
140 PRINT
150 PRINT JIS$(KMID$(K$,2,1))
160 END

RUN
文字を入力? 漢字

3441
OK
```

KACNV$

【ケー・エイ・コンバート・ダラー】

Kanji Ank CoNVert
2バイト文字を ANK 文字(1バイト文字)に変換(convert)すること。

機 能　文字列中に含まれる2バイト文字を1バイトの英数字に変換します。

書 式　KACNV$(文字列)

使用例　A$=KACNV$(K$)　←文字列 K$に含まれる2バイト文字の英数字を1バイト文字に変換します。

解 説
- 指定した**文字列**に含まれる2バイト文字を、1バイト文字に変換します。2バイト文字を示す KI/KO コードは取り除きます。
- 2バイト文字に対応する1バイト文字が存在しない場合は Illegal function call(違法関数呼び出し)エラーになります。
- 1バイト文字は変化しません。

参 照　AKCNV$(1バイト文字を2バイト文字に変換)

プログラム例

```
100 ' KACNV$
110 ' ---  2バイト文字を1バイト文字に変換する  ---
120 K$="ＡＢＣアイウ１２３ABC"
130 PRINT KACNV$(K$)
140 END

RUN
ABCｱｲｳ123ABC
OK
```

K

KEXT$

【カンジ・エクストラクト・ダラー】

Kanji EXTract
extract は取り除くの意味。ここでは文字列を抜き出すこと。

機能　文字列の中から、1バイト文字だけ、あるいは2バイト文字だけを抜き出します。

書式　KEXT$(文字列, 機能)

使用例　A$=KEXT$(B$, 0)　←文字列 B$ の中の1バイト文字を抜き出して、A$ に代入します。

解説

- **機能**で指定した種類の文字を、指定した**文字列**から抜き出します。**機能**は0または1で指定します。

機能	意味
0	英数字、カタカナなどの1バイト文字を抜き出します。
1	漢字などの2バイト文字を抜き出します。ただし KI/KO コードは抜き出さないため注意してください。

- 指定した**文字列**に、**機能**で指定した種類の文字が存在しない場合は、空文字列("")を返します。

プログラム例

```
100 ' KEXT$
110 ' ---  日本語文字を抜き出す  ---
120 K$="ABC日本語123ひらがなｱｲｳカタカナ"
130 KI$=CHR$(&H1B)+CHR$(&H4B)
140 KO$=CHR$(&H1B)+CHR$(&H48)
150 A$=KI$+KEXT$(K$,1)+KO$
160 PRINT A$
170 END

RUN
日本語ひらがなカタカナ
OK
```

KEY

【キー】

KEY
キー(鍵盤)のこと。ここではファンクションキーを制御すること。

機 能　ファンクションキーに任意の文字列を設定します。

書 式　**KEY　キー番号，文字列**

使用例　KEY 1，"SAVE"+CHR$(34)　←ファンクションキー f・1 に SAVE" を設定します。

解 説

- **キー番号**で指定したファンクションキーに、**文字列**を設定します。
- **キー番号**には1から10までの数値を指定し、それぞれファンクションキー f・1 から f・10 に対応します。
- **文字列**には、コントロールコードを含む1バイト文字を15文字以内で設定することができます。コントロールコードを設定するにはCHR$関数を使用します。また日本語文字列を設定することはできません。
- **文字列**として空文字列("")を指定すると、ファンクションキーとしての機能がなくなります。

参 照　CONSOLE(ファンクションキーの画面への表示)
KEY LIST(ファンクションキーの内容を表示)

プログラム例

```
KEY 1,"load "+CHR$(34)
KEY 5,"run"+CHR$(13)
```

K

KEY LIST

【キー・リスト】

KEY LIST
ここでは key はファンクションキーのこと。ファンクションキーの内容を出力(list)すること。

機　能　ファンクションキーに設定されている内容を一覧で画面に表示します。

書　式　KEY LIST

使用例　KEY LIST　←ファンクションキーの内容を画面に表示します。

解　説
• ファンクションキーに設定している文字列を以下の形式で画面に表示します。

F 1	load "	F 6	save "
F 2	auto	F 7	key
F 3	go to	F 8	print
F 4	list	F 9	edit .$^{C}{}_{R}$
F 5	run$^{C}{}_{R}$	F 10	cont$^{C}{}_{R}$

• CONSOLE 文で画面の最下行に常時ファンクションキーのリストを表示することができます。

CONSOLE,, 1	ファンクションキーの表示
CONSOLE,, 0	ファンクションキーの消去

参　照
CONSOLE(ファンクションキーの表示)
KEY(ファンクションキーの設定)

KEY ON
KEY OFF
KEY STOP

【キー・オン】
【キー・オフ】
【キー・ストップ】

KEY
ここでは key はファンクションキーのこと。これからファンクションキーに関する制御を行うこと。

機 能　ファンクションキーによる割り込み処理ルーチンの実行を許可、禁止、停止します。

書 式　KEY [(キー番号)] ｜ON｜OFF｜STOP｜

使用例

KEY ON	←割り込み処理ルーチンを実行します。
KEY OFF	←割り込み処理ルーチンの実行を禁止します。
KEY STOP	←割り込み処理ルーチンの実行を停止します。

K

解 説

- **キー番号**で指定したファンクションキーを押すと実行する割り込み処理ルーチンの制御を行います。**キー番号**は、ファンクションキー f･1 から f･10 に対応する1から10までの数値です。**キー番号**を省略すると、すべてのファンクションキーが対象になります。
- 割り込み処理ルーチンは ON KEY GOSUB 文で定義します。
- **KEY ON** は割り込み処理ルーチンの実行を許可します。この命令を実行後は、指定したファンクションキーを押すと ON KEY GOSUB 文で定義した割り込み処理ルーチンを実行します。
- **KEY OFF** は割り込み処理ルーチンの実行を禁止します。この命令を実行後は、指定したファンクションキーを押しても割り込みは発生せず、ファンクションキー本来の動作をします。プログラム終了時には必ず、KEY OFF を実行してください。
- **KEY STOP** は割り込み処理ルーチンの実行を停止します。この命令を実行後は、指定したファンクションキーが押されたことだけを記憶し、割り込み処理ルーチンは実行しません。このあと、KEY ON を実行すると、先ほどファンクションキーが押されたことによって割り込み処理ルーチンを実行します。

参 照　ON KEY GOSUB(ファンクションキーによる割り込み処理ルーチンの定義)

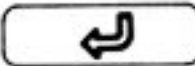
KILL

【キル】

KILL
殺す、消すということから、ファイルを削除すること。

機能　ディスク上の指定したファイルを削除します。

書式　KILL　ファイル指定子

使用例　KILL "DUST. DAT"　←ドライブ1のDUST. DATというファイルを削除します。

解説
- **ファイル指定子**で指定したファイルを削除します。
- 削除しようとするファイルは、クローズしていなければなりません。オープンしているファイルを指定すると、File already open(ファイルがすでにオープンされている)エラーになり、ファイルを削除できません。
- プロテクトセーブしたファイルもKILL文で削除することができます。
- SET文でファイル属性を書き込み禁止にすると、KILL文でそのファイルを削除することはできません。

KINPUT

【ケー・インプット】

Kanji INPUT
漢字(日本語文字)を入力(input)すること。

機能　自動的に、日本語入力モードに切り換えて、キーボードからデータを入力します。

書式　KINPUT　文字変数名

使用例　KINPUT A$　　←日本語入力モードに入り、キーボードからA$に日本語文字列を読み込みます。

解説

- KINPUT 文を実行すると自動的に日本語入力モードに切り換わり、キーボードからのデータ入力待ちになります。INPUT 文のように疑問符(?)と空白を表示します。
- 日本語入力の方法にしたがって日本語文字列の入力を行い、[↵]キーを押すと、指定した**文字変数**にデータを読み込みます。このとき、自動的に日本語入力モードを抜けます。
- 一つの KINPUT 文に一つの**文字変数**しか指定できません。
- 日本語入力は、必ず画面の偶数桁から表示していきます。LOCATE 文で奇数桁を指定しても、その桁数＋1から表示します。

プログラム例

```
100 ' KINPUT
110 '----  自動的に日本語入力モードに入る  ----
120 CLS
130 LOCATE 0,0 : PRINT "日本語を入力してください。"
140 LOCATE 30,0: KINPUT A$
150 LOCATE 0,2 : PRINT "入力した日本語は「";A$;"」です。"
160 END
```

```
日本語を入力してください。    ?  中国の歴史


入力した日本語は「中国の歴史」です。
OK
```

K

KINSTR

【カンジ・インストリング】

Kanji IN STRing
string は一連とか一列の意味。転じて、文字列のこと。in は～のなかの、～についての意味の接頭語。

機 能　2バイト文字を含む文字列の中から指定した文字列を探し出し、見つかればその文字列の開始位置を知らせます。

書 式　KINSTR([開始位置,] 探索される文字列, 見つける文字列)

使用例　A=KINSTR(10, B$, C$)　　←B$の10文字目からC$を探索します。

解 説

- **探索される文字列**の中を、**開始位置**から**見つける文字列**を探します。見つかれば、その位置を**探索される文字列**の左端からの文字数で知らせます。もし見つからない場合は、0になります。
- **開始位置**を**探索される文字列**の左端からの文字数で指定します。このとき2バイト文字も1文字として数えます。またKI, KOコードも1文字として文字数に含めます。省略すると文字列の最初から探します。
- **見つける文字列**に空文字列("")を指定すると、開始位置と同じ値を返します。
- 同じように文字列の位置を求めるINSTR関数は、単純に文字列のバイト数で指定します。

例
```
A$="AB 漢字カナまじり"
PRINT KINSTR(A$, "まじり")
 10
OK
```

文字数	1	2	3	4	5	6	7	8	9	10	11	12	13
文字列	A	B	KI	漢	字	KO	カ	ナ	KI	ま	じ	り	KO
バイト	1	2	3 4	5 6	7 8	9 10	11	12	13 14	15 16	17 18	19 20	21 22

プログラム例

```
100 ' KINSTR
110 ' ---  名前を探す  ---
120 CLS
130 M$="中森明菜,石川秀美,菊池桃子,本田美奈子"
140 INPUT "お名前は ";N$
150 F=KINSTR(M$,N$)
160 PRINT
170 IF F=0 THEN PRINT "はじめまして ";N$;" さん"
180 IF F<>0 THEN PRINT "ようこそ! ";N$;" さん"
190 END

RUN
お名前は ? 本田美奈子

ようこそ! 本田美奈子 さん
OK
```

KLEN

【カンジ・レングス】

Kanji LENgth
漢字(2バイト文字)の長さ(length)のこと。

機能 2バイト文字を含む文字列の長さを与えます。

書式 KLEN(文字列 [, 機能])

使用例 A=KLEN(X$) ←文字列の長さを変数Aに代入します。

解説

- 指定した**文字列**の長さを1バイト文字、2バイト文字をそれぞれ1文字として数えます。
- **機能**は数える文字の種類を0から5までの整数で指定します。

機能	意味
0	文字列全体の文字数。2バイト文字、KI/KOコードもそれぞれ1文字として数えます。
1	英数字、カタカナなどの1バイト文字の文字数。
2	2バイト文字の文字数。KI/KOコードは含みません。
3	2バイト文字のうち、漢字などの全角文字の文字数。
4	2バイト文字のうち、英数字などの半角文字の文字数。
5	KI/KOコードの文字数。

- 同じように文字列の長さを与えるLEN関数は、単純に文字列のバイト数を返します。

参照 LEN(文字列の長さを与える)

プログラム例

```
100 ' KLEN
110 ' ---   文字数を数える   ---
120 A$="ＡＢＣABCABCいろはｲﾛﾊ漢字"
130 PRINT A$
140 PRINT
150 PRINT "総文字数 =";KLEN(A$,0)
160 PRINT "英数カナ =";KLEN(A$,1)
170 PRINT "日本語    =";KLEN(A$,2)
180 PRINT "全角文字 =";KLEN(A$,3)
190 PRINT "半角文字 =";KLEN(A$,4)
200 PRINT "KI/KOｺｰﾄﾞ=";KLEN(A$,5)
210 END

RUN
ＡＢＣABCABCいろはｲﾛﾊ漢字

総文字数 = 23
英数カナ = 6
日本語   = 11
全角文字 = 8
半角文字 = 3
KI/KOｺｰﾄﾞ= 6
OK
```

K

KMID$

【カンジ・ミッド・ダラー】

Kanji MIDdle
日本語文字列(漢字)の中(middle)から新しく文字列を作成します。

機　能　2バイト文字を含む文字列の中から、指定する部分の文字列を抜き出します。

書　式　KMID$(文字列，初めの位置［，文字数］)

使用例　C$=KMID$("100 東京都 101"，4，5)　←文字列"100 東京都 101"から"東京都"を抜き出します。

解　説
- 指定した**文字列**の中から、**初めの位置**で指定した位置から**文字数**で指定した文字列を抜き出します。実行後、指定した**文字列**の内容は変化しません。
- **初めの位置**、**文字数**とも1バイト文字、2バイト文字およびKI/KOコードを1文字として数えた値です。
- **初めの位置**が、文字列の文字数より大きい場合は、空文字列("")を返します。
- **文字数**を省略したり、**初めの位置**からの文字列の文字数が**文字数**で指定した値より短い場合は、**初めの位置**から文字列の終わりまでを抜き出します。
- 同じように文字列を抜き出すMID$関数は、単純にバイト数で計算した値を用います。

参　照　MID$(文字列の抜き出し)

プログラム例

```
100 ' KMID$
110 ' ---  文字列の中から、文字を抜き出す  ---
120 A$="ABC漢字とDEF日本語"
130 B$=KMID$(A$,4,5)+KMID$(A$,12,5)
140 C$=KMID$(A$,1,3)+KMID$(A$,9,3)
150 PRINT A$;" ---> ";B$;" + ";C$
160 END

RUN
ABC漢字とDEF日本語 ---> 漢字と日本語 + ABCDEF
OK
```

KNJ$

【カンジ・ダラー】

KaNJi
ここでは2バイトのコードを漢字に変換すること。

機能 JIS漢字コードで与えた4桁の文字列を、対応する漢字に変換します。

書式 KNJ$(文字列)

使用例 PRINT KNJ$("1B4B")+KNJ$("3441")+KNJ$("1B48")←漢字コード3441を"漢"に変換し画面に表示します。

解説

- JIS漢字コードで与えた**文字列**を、対応する漢字に変換します。指定した**文字列**に対応する2バイト文字が存在しない場合は、Illegal function call(違法関数呼び出し)エラーになります。
- **文字列**は、JIS漢字コードの範囲あるいはKI/KOコードでなければなりません。
- 漢字を表示するには、前後に漢字コードを示すKI/KOコードが必要です。
- 逆に、2バイト文字の文字コードを調べる関数としてJIS$関数があります。

参照 JIS$(2バイトの漢字コードを与える)

K

プログラム例

```
100 ' KNJ$
110 ' ---  漢字コードを日本語文字へ変換する  ---
120 KI$="1B4B" : KO$="1B48"
130 FOR I=0 TO 9
140   READ K$
150   PRINT KNJ$(KI$)+KNJ$(K$)+KNJ$(KO$);
160 NEXT I
170 END
180 DATA 3440,3441,3442,3443,3444,3445,3446,3447,3448,3449

RUN
汗漢澗潅環甘監看竿管
OK
```

KPLOAD

【ケー・ピー・ロード】

Kanji Pattern LOAD
loadは積むという意味。ここでは、ユーザーが定義した文字パターンを登録すること。

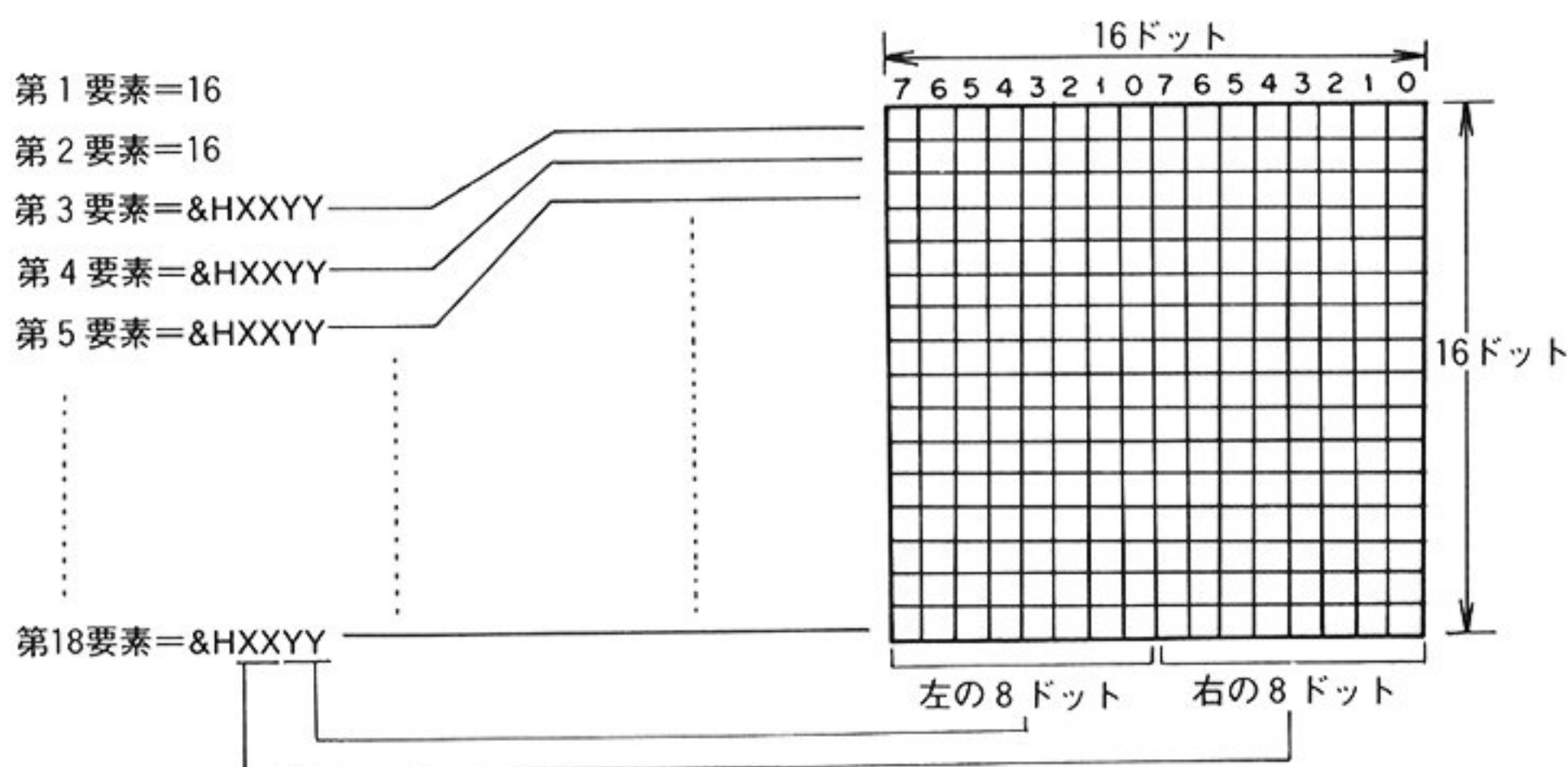
機　能　自由に作成した文字パターンを、指定した漢字コードに登録します。

書　式　KPLOAD　漢字コード，整数型配列名

使用例　KPLOAD &H7621, KPN%　←漢字コード&H7621に、整数型配列KPN%の文字パターンを登録します。

解　説
- 指定した**漢字コード**に、あらかじめ1次元の**整数型配列**に設定してある文字パターンを登録します。以後、この**漢字コード**の文字を表示すると、ここで設定した文字パターンを表示します。
- **漢字コード**の指定できる範囲は次のとおりです。

  &H7621　～　&H767E

  &H7721　～　&H777E
- 整数型配列には、次の形式で文字パターンを設定します。

第3要素から第18要素までに、実際の文字パターンのドットイメージを16進数で指定します。
- 配列の添字の最小値が0であれば、0から、1であれば1から順にデータを設定します。設定するデータの数は18個であるため、あらかじめDIM文で配列の宣言をしておく必要があります。

KTYPE

【ケー・タイプ】

Kanji TYPE
漢字を含む文字の種類(type)のこと。

機能　2バイト文字を含む文字列中の指定位置の文字の種類を与えます。

書式　KTYPE(文字列, 位置)

使用例　T=KTYPE(A$, 3)　←文字列 A$の3文字目の文字の種類を変数Tに代入します。

解説

• 指定した**文字列**の、先頭からの**位置**で指定した文字の種類を以下の値で知らせます。

関数値	意味
0	1バイト文字
1	2バイト文字の全角文字
2	2バイト文字の半角文字
3	KI コード
4	KO コード

• **位置**は文字列の先頭からの位置を示し、1バイト文字、2バイト文字および KI/KO コードを1文字として数えた値です。
• 指定した**位置**に文字が存在しない場合は、Illegal function call(違法関数呼び出し)エラーになります。

例：• 指定した文字列が空文字列("")の場合
　　• 指定した位置が0の場合
　　• 指定した位置が文字列の長さより大きい場合

プログラム例

```
100 ' KTYPE
110 '----  文字のタイプを調べる  ---
120 A$="ABC漢字DEF日本ｲﾛﾊ"
130 PRINT " A  B  C  KI 漢 字 D  E  F  日 本 KO ｲ  ﾛ  ﾊ"
140 PRINT " :  :  :  :  :  :  :  :  :  :  :  :  :  :  :"
150 FOR I=1 TO 15
160   PRINT KTYPE(A$,I);
170 NEXT I
180 END

RUN
 A  B  C  KI 漢 字 D  E  F  日 本 KO ｲ  ﾛ  ﾊ
 :  :  :  :  :  :  :  :  :  :  :  :  :  :  :
 0  0  0  3  1  1  2  2  2  1  1  4  0  0  0
OK
```

LEFT$

【レフト・ダラー】

LEFT
leftは左側のこと。

機　能　文字列の左から指定した文字数分の文字列を返します。

書　式　LEFT$(文字列, 文字数)

使用例　A$=LEFT$(B$, 5)　←文字列B$の左から5文字を取り出して、A$に代入します。

解　説
- 指定した**文字列**の左から、**文字数**で指定するバイト数の文字列を返します。
- **文字数**には、0から255までの範囲で、取り出す文字列のバイト数を指定します。**文字数**が文字列全体のバイト数より大きいときは、その文字列のすべてを取り出します。また、長さが0であれば空文字列("")を返します。
- もとの**文字列**の内容は変化しません。

参　照　MID$(文字列の指定位置から文字列を取り出す)
RIGHT$(文字列の右側から文字列を取り出す)

プログラム例

```
100 ' LEFT$
110 ' ---  左からの文字列を読み取る  ---
120 A$="東京都千代田区神田"
130 FOR I=2 TO 20 STEP 2
140   B$=LEFT$(A$,I)
150   PRINT B$
160 NEXT I
170 END

RUN

東
東京
東京都
東京都千
東京都千代
東京都千代田
東京都千代田区
東京都千代田区神
東京都千代田区神田
OK
```

LEN

【レングス】

LENgth
length は長さの意味。

機 能　文字列の長さをバイト数で返します。

書 式　LEN(文字列)

使用例　A=LEN(B$)　←文字列 B$ の長さ(バイト数)を A に代入します。

解 説
- 指定した**文字列**の長さをバイト数で返します。
- 2バイト文字および KI/KO コードはそれぞれ2バイトとして数えます。これに対して KLEN 関数は、文字列の長さを文字数として数える関数です。

参 照　KLEN(文字列の長さを文字数で与える)

プログラム例

```
100 ' LEN
110 ' ---   文字列の長さ   ---
120 A$="ABC日本語DEF ｲﾛﾊ"
130 PRINT LEN(A$)
140 END

RUN
 20
OK
```

LET

【レット】

LET
ある状態にするという意味から、変数に数値や文字列などを代入すること。

機能　右辺の式の結果を左辺の変数に代入します。

書式　[LET]　変数名＝ |数式
文字列|

使用例　LET A=10　←数値10を変数Aに代入します。

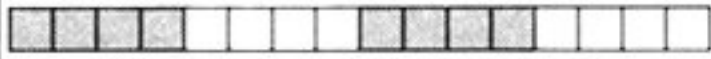
解説
- 右辺の**数式**や**文字列**を左辺の**変数**に代入します。このとき、右辺が数式であれば左辺は数値変数を、右辺が文字列であれば左辺は文字変数を指定しなければいけません。これが一致しない場合は、Type mismatch(型が異なっている)エラーになります。
- 数式と数値変数の型が異なる場合は、数値変数の型に型変換して代入します。
- LET は省略できます。

LINE

【ライン】

LINE
線を引くこと。したがって画面に線を描くこと。

機能　画面に線や四角形を描きます。

書式　LINE [(Wx1, Wy1)] −(Wx2, Wy2) [, [描画色] [, [B | BF] [, [ラインスタイル | 塗色 | タイルストリング]]]]

使用例

```
LINE(50, 100)-(200, 150)        ←線を描きます。
LINE(50, 100)-(200, 150),, B    ←四角形を描きます。
```

解説

- ワールド座標系の座標(**Wx1, Wy1**)と(**Wx2, Wy2**)の2点を結ぶ直線を描きます。(**Wx1, Wy1**)を省略した場合は最終参照座標を指定したことになります。また、それぞれの座標の前にSTEPを付けて、相対座標で指定することができます。
- **描画色**には、描く線の色をパレット番号で指定します。省略するとCOLOR文で設定したグラフィック画面の前景色で描きます。
- **描画色**に続けて**B**または**BF**の指定を行うと次のような四角形を描くことができます。

B指定　座標(Wx1, Wy2)と(Wx2, Wy2)の2点を対角とする四角形を描きます。

BF指定　座標(Wx1, Wy2)と(Wx2, Wy2)の2点を対角とする四角形を描き、その内部を塗りつぶします。

- **ラインスタイル**は、どのような線を描くのかを16進数の&H0000から&HFFFFの値で指定します。これは、画面上の16ドットを2バイトの各ビットに一対一に対応したデータです。このデータを繰り返し表示し線を描きます。

例

	ドットと線	ビット	16進数
実線	■■■■■■■■■■■■■■■■	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	&HFFFF
荒い破線	■■■■□□□□■■■■□□□□	1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0	&HFOFO
細い破線	■□■□■□■□■□■□■□■□	1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0	&HAAAA

ビットが1のとき：画面上にドットを表示する。
ビットが0のとき：画面上にドットを表示しない。

&H0000から&HFFFFの数値であれば10進数で指定しても構いません。ただし、このように16ドットに対応させて考える場合は、16進数で表現したほうが便利です。

L

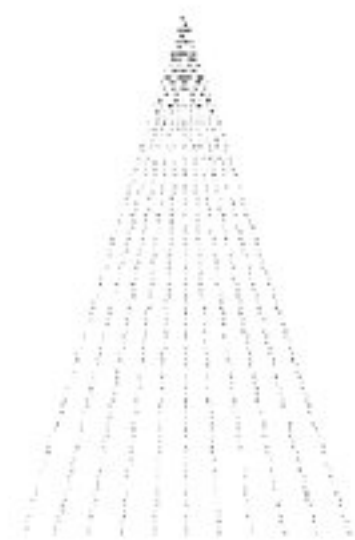
・**ラインスタイル**を省略した場合は、&HFFFF を指定したことになり、実線を描きます。

・**塗色**と**タイルストリング**は **BF** 指定を行ったときのみ意味を持ちます。**塗色**には、パレット番号を指定し、指定したパレット番号で四角形の内部を塗りつぶします。また**タイルストリング**を指定すれば四角形の内部を模様で塗りつぶします。**塗色**、**タイルストリング**を省略した場合は、四角形を描いた色と同じ色で内部を塗りつぶします。**タイルストリング**については PAINT 文を参照してください。

・LINE 文を実行すると、最終参照座標は(Wx2, Wy2)に移動します。

参　照　PAINT(指定した領域の塗りつぶし)

プログラム例

```
100 ' LINE
110 ' ---   ＬＩＮＥの使用例  ---
120 SCREEN 0,0,0,1 : CLS 3
130 '
140 FOR X=0 TO 100 STEP 10
150   LINE(50,10)-(X,100),2
160 NEXT
170 FOR I=0 TO 45 STEP 5
180   LINE(150+I,10+I)-(250-I,100-I),3,B
190 NEXT
200 LINE(300,10)-(400,100),4,BF
210 'LINE(450,10)-(550,100),5,BF,1
220 '
230 LINE(0,120)-(399,120),2,,&H1111
240 LINE(0,130)-(399,130),3,,&HAAAA
250 LINE(0,140)-(399,140),4,,&H7777
260 LINE(0,150)-(399,150),5,,&HF1F1
270 LINE(0,160)-(399,160),6,,&H7FF7
280 LINE(0,170)-(399,170),7,,&HF99F
290 END
```

LINE INPUT

【ライン・インプット】

LINE INPUT
lineの1行という意味から、1行全体を読み込む(input)こと。

機能　キーボードから〔↵〕キーを押すまでを1つのデータとして文字変数に読み込みます。

書式　LINE INPUT　["入力メッセージ" ｛;｜,｝] 文字変数

使用例　LINE INPUT "ADDRESS：" ; A$　←例えば住所としてChiyodaku, Tokyoなどのようにカンマを入力することができます。

解説
- LINE INPUT文を実行すると **"入力メッセージ"** を表示し、キーボードからのデータ入力待ちになります。省略した場合は何も表示しません。
- キーボードからデータを入力し〔↵〕キーを押すと、入力したデータを**文字変数**に読み込みます。
- **入力メッセージ**は入力待ち状態になる前に画面に表示する文字列で、文字定数を指定します。文字変数は使用できません。
- INPUT文とは異なり、入力メッセージを省略しても疑問符(?)を表示しません。

参照
INPUT(キーボードからのデータ入力)
INPUT$(キーボードからのデータ入力)
KINPUT(日本語データ入力)

プログラム例

```
100 ' LINE INPUT
110 '----  文字列単位の入力  ----
120 LINE INPUT "NAME    : ";N$
130 LINE INPUT "ADDRESS : ";A$
140 PRINT
150 PRINT N$;" : ";A$
160 END

RUN
NAME    : Barry Gibson
ADDRESS : 186 Lincoln St., Boston, MA 2111

Barry Gibson : 186 Lincoln St., Boston, MA 2111
OK
```

LINE INPUT#

【ライン・インプット・シャープ】

LINE INPUT
lineの1行という意味から、ファイルから1行全体を読み込む(input)こと。

機　能　シーケンシャルファイルから、行単位(最大255バイト)で文字列を文字変数に読み込みます。

書　式　LINE INPUT [#]ファイル番号, 文字変数

使用例
```
100  OPEN "DATA" FOR INPUT AS #1
110  LINE INPUT#1, A$
```
←シーケンシャルファイルのデータの1行分をA$に読み込みます。

解　説
- **ファイル番号**で指定したファイルから、CRコードまでを1行のデータとして**文字変数**に読み込みます。
- CRコードが現れない場合は、最大255バイトを一つのデータとして処理します。

参　照　INPUT#(シーケンシャルファイルからデータを読む)

プログラム例

```
100 ' LINE INPUT#
110 ' ---  シーケンシャルファイルから1行単位でデータを読み込む  ---
120 OPEN "2:DATA1" FOR INPUT AS #1
130 IF EOF(1) THEN GOTO 170
140   LINE INPUT #1,NM$
150   PRINT NM$
160 GOTO 130
170 CLOSE #1
180 END
```

LINE INPUT WAIT

【ライン・インプット・ウェイト】

LINE INPUT WAIT
wait は待つこと。したがって待ち時間を制限した LINE INPUT文。

機能 キーボードから変数にデータを入力します。その際に、その入力待ち時間を制限します。

書式 LINE INPUT WAIT 待ち時間, ["入力メッセージ" |;|,|] 文字変数

使用例 LINE INPUT WAIT 300, "NO. =";N$ ←N$にデータを読み込む際に30秒だけ待ちます。

解説

- キーボードから LINE INPUT 文と同じように、**変数**にデータを読み込みます。その際に、**待ち時間**で指定した時間だけ入力を待ちます。**待ち時間**の単位は0.1秒です。
  待ち時間には 0 を除いた－32768から65535までの整数を指定します。ただし負の数を指定した場合は65536に指定した値を加算した結果を指定したことになります。例えば **LINE INPUT WAIT －1, A$** と **LINE INPUT WAIT 65535, A$** は同じ時間だけ待ちます。
- この文の後ろにマルチステートメントとして他の命令文がある場合、待ち時間内にデータを入力したときだけ後の命令文を実行します。入力を行わなかったときは、後の文を実行せずに、次の行を実行します。
- 待ち時間の制限があることを除けば、LINE INPUT 文と同じ機能です。

参照 LINE INPUT(キーボードからのデータ入力)

LIST
LLIST

【リスト】【エル・リスト】

LIST
表、一覧表の意味。すなわち、項目を並べること。

LIST to Line printer
listの内容をプリンタに出力すること。

機能 メモリ中のプログラムの内容を画面に表示したり、プリンタに出力します。

書式
LIST　[初めの行番号] [－ [終わりの行番号]]
LLIST　[初めの行番号] [－ [終わりの行番号]]

使用例

LIST	←プログラムの内容すべてを表示します。
LIST 100－200	←100行から200行までを表示します。

解説

- LISTコマンドは、プログラムの内容を画面に表示します。
- LLISTコマンドは、プログラムの内容をプリンタに出力します。
- **初めの行番号**と**終わりの行番号**で出力する範囲を指定します。それぞれの行番号の指定は省略することができ、省略した場合は次のような行番号を指定したことになります。

書　式	意　味
LIST	プログラム全体
LIST 初めの行番号－終わりの行番号	初めの行番号から終わりの行番号まで
LIST 初めの行番号	初めの行番号だけ
LIST 初めの行番号－	初めの行番号からプログラムの最後まで
LIST －終わりの行番号	プログラムの最初から終わりの行番号まで

- **行番号**にピリオド(.)を指定すると、直前に参照した行を指定したことになります。
- 指定した**行番号**がプログラム中に存在しない場合はその指定範囲内の行を表示します。
- LIST, LLISTコマンドの実行後は、BASICのコマンド入力待ちになります。
- 出力を途中で停止するには [CTRL] + [S] キーを押します。再開するには任意のキーを押します。また途中で終了するには [CTRL] + [C] または [STOP] キーを押します。

LOAD

【ロード】

LOAD
荷物を積む意味から、プログラムをディスクなどからメモリに読み込む(ロードする)こと。

機能

プログラムを指定した装置からメモリに読み込みます。また、そのプログラムを引き続き実行することもできます。

書式

LOAD　ファイル指定子 [, R]

使用例

```
LOAD "DEMO. BAS"
```

←DEMO. BASというファイルを読み込みます。

解説

- **ファイル指定子**で指定したプログラムファイルをメモリに読み込みます。これをプログラムをロードするといいます。
- **R**を指定すると、ロードしたプログラムを引き続き実行します。これは**RUN　ファイル指定子**と同じ機能になります。
- LOADコマンドは、指定したプログラムファイルを読み込む前に、メモリ上のプログラムや変数の消去を行います。また**R**指定がない場合は、オープンしているすべてのファイルをクローズします。**R**指定をした場合は、ファイルのクローズは行いません。

参照

RUN(プログラムの実行)

LOC

【エル・オー・シー/ロック】

LOCation
位置、場所の意味から、ファイル中の位置を求めること。

機　能　ファイル中で、現在入出力を行っている位置を与えます。

書　式　LOC([#]ファイル番号)

使用例

```
100  OPEN "COM :" AS #1
200  IF LOC(1)>255 THEN PRINT #1, CHR$(19)
```

←受信バッファにデータを255バイト以上受信したらXONコードを送信します。

解　説

- **ファイル番号**で指定したファイルがどのような種類のファイルかによってLOC関数の返す値は異なります。
- 指定したファイルがシーケンシャルファイルの場合
  直前に入出力を行ったデータの位置をファイルの先頭から256バイト単位の位置で返します。
- 指定したファイルがランダムファイルの場合
  直前にGET文あるいはPUT文で指定したレコード番号を返します。
- 指定したファイルが通信回線の場合
  受信バッファ内の受信データのバイト数を返します。
- 指定したファイルがキーボードの場合
  キーボードバッファ内のデータのバイト数を返します。

参　照　EOF(ファイルの終わり)
LOF(ファイルの大きさ)

プログラム例

```
100 ' LOC
110 ' ---  ファイル中の現在位置  ---
120 OPEN "2:DATA" AS #1
130 FIELD #1,10 AS A$,15 AS B$
140 FOR R=1 TO LOF(1)-1
150   PRINT "No.:";LOC(1)
160   GET #1,R
170   PRINT "      ";A$,B$
180 PRINT
190 NEXT R
200 REC=LOC(1)
210 PRINT:PRINT "読み出したデータ数";REC
220 CLOSE #1
```

LOCATE

【ロケイト】

LOCATE
位置を定めることから、ここではカーソルの位置を指定すること。

機 能　テキスト画面のカーソル位置や表示の有無を指定します。

書 式　LOCATE　[桁位置] [, [行位置] [, [カーソルスイッチ]]]

使用例　LOCATE 10, 15　←カーソル位置を16行11桁へ移動します。

解 説

- **桁位置**、**行位置**で指定した位置へ、カーソルを移動します。**桁位置**、**行位置**は画面の左上隅を(0, 0)とするキャラクタ座標の位置で指定します。
- **桁位置**を省略した場合は0が、**行位置**を省略した場合は、現在のカーソルがある位置の行を指定したことになります。
- **桁位置**を奇数桁に指定して日本語などの2バイト文字の表示を行うと、正しく表示できません。
- **カーソルスイッチ**はカーソルの表示状態を決めるもので0または1の数値を指定します。

カーソルスイッチ	意　味
0	カーソルを表示しません。
1	カーソルを表示します。

L

LOF

【エル・オー・エフ】

Length Of File
length とは長さのこと。すなわちファイルの大きさ。

機能　ファイルの大きさを与えます。

書式　LOF([#]ファイル番号)

使用例

```
100 OPEN "TEST" AS #1
200 A=LOF(1)
```

←ランダムファイル TEST の最大のレコード番号、すなわちファイルの大きさをAに代入します。

解説

- **ファイル番号**で指定したファイルの大きさを示します。ただし**ファイル番号**で指定したファイルがどのような種類のファイルかによって LOF 関数の返す値は異なります。
- 指定したファイルがシーケンシャルファイルの場合
  ファイルの大きさを使用しているセクタ(256バイト)数で返します。
- 指定したファイルがランダムファイルの場合
  最大のレコード番号
- 通信回線
  通信バッファ内の未使用空間のバイト数を示します。受信バッファの大きさは256バイトです。

参照

EOF(ファイルの終わり)
LOC(ファイル中の現在位置)

LOG

【ログ】

LOGarithm
ここでは自然対数 log の意味。

機　能　自然対数(e≒2.71828を底とする対数)を計算します。

書　式　LOG(数式)

使用例　A=LOG(10)　←10の自然対数を計算してAに代入します。

解　説
- ( )内の**数式**の e を底とする自然対数を計算します。
- **数式**が倍精度実数のときは、結果も倍精度で計算した値になります。他の場合は、単精度で計算した値になります。
- log 関数の逆関数である e^x は EXP 関数で計算できます。
- 10を底とする常用対数 $\log_{10}x$ は底の変換により次のように定義できます。

```
DEF FNLOG10(X)=LOG(X)/LOG(10#)
```

参　照　EXP(指数関数)

L

LPOS

【ライン・ポジション】

Line POSition
ここでは line はプリンタの意味。したがって、プリンタの位置を求めること。

機　能　プリンタバッファ内に保持しているプリンタヘッドの位置を示します。

書　式　LPOS(引数)

使用例　A=LPOS(0)　　←プリンタヘッドの位置を返します。

解　説
- **引数**はダミーであり意味を持ちませんが、形式上数値定数などを記述します。
- LPOS 関数は、論理的なプリンタヘッドの位置を返すもので、物理的なプリンタヘッドの位置を返すものではありません。
- プリンタバッファは、WIDTH LPRINT 文で設定する1行の出力文字数のことです。LPOS 関数は、このプリンタバッファに対して、何文字出力したかを与えるものです。

参　照　WIDTH LPRINT(プリンタ出力の一行の文字数設定)

LSET
RSET

【エル・セット】【アール・セット】

Left SET
データを左詰めでセットすること。

Right SET
データを右詰めでセットすること。

機能　ランダムファイルバッファにデータをセット(配置)します。

書式　LSET　文字変数＝文字列
RSET　文字変数＝文字列

使用例

```
LSET A$=X$        ←X$ を左詰めで A$ に配置します。
RSET B$=Y$        ←Y$ を右詰めで B$ に配置します。
```

解説

- ランダムファイルバッファにデータを書き込みます。
- **文字変数**には、あらかじめ FIELD 文で定義したバッファの変数名を指定します。
- LSET 文は左詰めに、RSET 文は右詰めに、指定した**文字列**を配置します。**文字列**の長さが、指定したバッファより大きい場合は、**文字列**の右側から文字を切り捨てます。逆に短い場合は、余ったバッファ内の領域には空白(" ")を設定します。
- ランダムファイルに対する入出力は文字列を単位に行います。したがって数値をランダムファイルバッファに設定するには、数値を文字列に変換する必要があります。数値から文字列への変換は、MKI$, MKS$, MKD$ 関数を使用します。

参照　MKI$/MKS$/MKD$(数値を文字列に変換)

MAP

【マップ】

MAP
地図の意味。ここでは座標変換を行うこと。

機 能　スクリーン座標をワールド座標に、逆にワールド座標をスクリーン座標に変換します。

書 式　MAP(座標，機能)

使用例　WX=MAP(SX，2)　←スクリーン座標のX座標をワールド座標のX座標に変換します。

解 説

- **座標**で指定したスクリーン座標あるいはワールド座標上の座標値を、**機能**で指定した座標に変換します。
- **機能**は0から3までの整数を指定し、変換の種類を示します。

機能	変換の意味
0	ワールド座標のX座標→スクリーン座標のX座標
1	ワールド座標のY座標→スクリーン座標のY座標
2	スクリーン座標のX座標→ワールド座標のX座標
3	スクリーン座標のY座標→ワールド座標のY座標

- ワールド座標の座標をスクリーン座標の座標に変換する際に、計算の結果が、スクリーン座標の範囲を超える場合もエラーは発生しません。
- ワールド座標の座標値をスクリーン座標の座標値に変換し、さらにその値をワールド座標の座標値に変換した場合、必ずしも、もとの値と同じになりません。

MERGE

【マージ】

MERGE
結合する、溶け込む意味から、2つのプログラムを合わせて1つにすること。

機能　ディスク上のプログラムを、メモリ中のプログラムに結合します。

書式　MERGE　ファイル指定子

使用例　MERGE "SUB.　BAS"　←プログラム SUB.　BAS をメモリ上のプログラムに結合します。

解説

- **ファイル指定子**で指定したプログラムファイルとメモリ中のプログラムを合わせて1つのプログラムにします。これをプログラムのマージといいます。
- 指定するプログラムファイルはアスキーセーブしたものでなければなりません。アスキーセーブしたファイルでない場合は Sequential I/O only（ファイルモードの誤り）エラーになります。
- メモリ中のプログラムと、ディスク中のプログラムに同じ行番号が存在する場合は、ディスク側の行番号の内容で置き換えを行います。
- MERGE コマンド実行後は BASIC のコマンド入力待ち状態になります。

参照

CHAIN（プログラムの連鎖）
LOAD（プログラムのロード）

MID$

【ミッド・ダラー】

MIDdle
中間とか中央の意味。ここでは文字列の中間に操作を行うこと。

機　能　文字列の一部を別の文字列で置き換えます。

書　式　MID$(対象文字列, 開始位置 [, 文字数])=置換文字列

使用例　MID$(X$, 1, 1)="A"　←文字列 X$ の初めの 1 文字を "A" に置き換えます。

解　説
- **対象文字列**の**開始位置**から**文字数(バイト数)**分だけ、**置換文字列**の先頭から**文字数**分で置き換えます。
- **文字数**を省略すると、**置換文字列**の文字数を指定したことになります。
- **文字数**が対象文字列の**開始位置**からの長さより長いときは、対象文字列の長さの範囲内で置き換えます。したがって、この操作によって**対象文字列**の長さが変化することはありません。
- **対象文字列**に空文字列("")は指定できません。

参　照　MID$(関数：文字列の中から指定部分を取り出す)

MID$

【ミッド・ダラー】

MIDdle
middle とは中間とか中央の意味。ここでは文字列の中から新しい文字列を作ること。

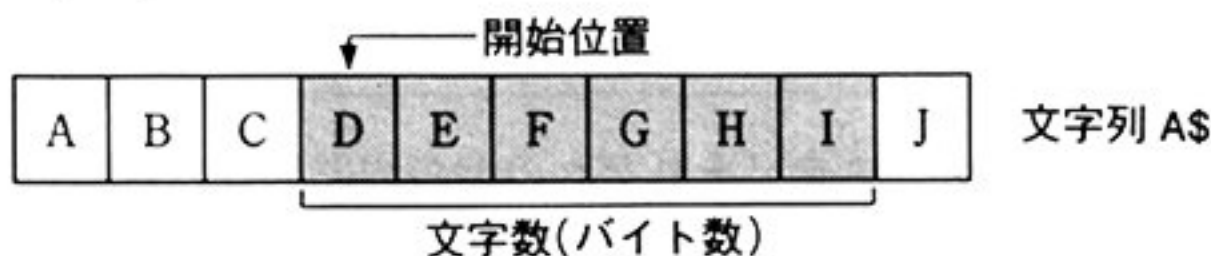
機 能 文字列の中から指定する文字数分の文字列を返します。

書 式 MID$(文字列, 開始位置 [, 文字数])

使用例 A$=MID$(B$, 3, 5) ←文字列 B$ の先頭から 3 バイト目の位置から 5 バイト分の文字列を取り出して A$ に代入します。

解 説

- 指定した**文字列**の中から、**開始位置**で指定した位置から**文字数**(バイト数)分だけの文字列を取り出します。指定した**文字列**の内容は変化しません。

例: A$="ABCDEFGHIJ"
　　B$=MID$(A$, 4, 6)

- **開始位置**は、文字列の先頭からのバイト数で 1 から255までの数値で指定します。
- **文字数**は、取り出す文字列の長さをバイト数で指定した値です。
- **文字数**を省略した場合や、**開始位置**から右側の文字列の長さが**文字数**で指定した値より短い場合は、**開始位置**から右側の文字列すべてを取り出します。
- **文字数**が 0 の場合や、**開始位置**の指定が文字列のバイト数より大きい場合は、空文字列("")を与えます。

参 照 LEFT$(文字列の左側から文字列を取り出す)
RIGHT$(文字列の右側から文字列を取り出す)

プログラム例

```
100 ' MID$
110 '----  文字列を抜き取る  ----
120 PRINT "現在の時刻  : ";
130 T$=TIME$
140 HOUR=VAL(MID$(T$,1,2))
150 IF HOUR<12 THEN PRINT "午前"; ELSE HOUR=HOUR-12 : PRINT "午後";
160 MIN=VAL(MID$(T$,4,2)) : SEC=VAL(MID$(T$,7,2))
170 PRINT USING "###";HOUR; : PRINT "時";
180 PRINT USING "###";MIN;  : PRINT "分";
190 PRINT USING "###";SEC;  : PRINT "秒"
200 END

RUN
現在の時刻  : 午後  6時 59分 26秒
OK
```

M

MKI$
MKS$
MKD$

【エム・ケイ・アイ・ダラー】
【エム・ケイ・エス・ダラー】
【エム・ケイ・ディ・ダラー】

MaKe Integer/Single/Double
それぞれ整数(integer)、単精度実数(single)、倍精度実数(double)を意味します。

機能 数値を内部形式の文字列に変換します。

書式
MKI$(整数)
MKS$(単精度実数)
MKD$(倍精度実数)

使用例

```
A$=MKI$(I%)      ←整数値を2バイトの文字列に変換
B$=MKS$(S!)      ←単精度実数を4バイトの文字列に変換
C$=MKD$(D#)      ←倍精度実数を8バイトの文字列に変換
```

解説

- **数値**データを、それぞれの型の内部形式に対応した文字列に変換します。**数値**の型と文字列の長さは次のとおりです。
- ランダムファイルは文字型のデータしか扱えません。このため数値を文字型に変換する必要があります。これらの関数は、数値をそれぞれの型の内部表現に応じた一定の長さの文字列に変換します。

数値の型	数値→文字列の関数	文字列→数値の関数	文字列の長さ
整数	MKI$	CVI	2バイト
単精度実数値	MKS$	CVS	4バイト
倍精度実数値	MKD$	CVD	8バイト

- この文字列を、もとの数値に変換するにはCVI, CVS, CVD関数を使用します。

参照 CVI/CVS/CVD(文字列から数値への変換)
日本語Disk BASICユーザーズマニュアル第11章　データの内部構造

MON

【モニタ】

MONitor
監視するなどの意味。直接メモリの中を操作すること。

機　能　BASICモードから機械語モニタモードに制御を移します。

書　式　MON

使用例　MON　　←機械語モニタモードに制御を移します。

解　説

• BASICモードから機械語モニタモードに制御を移します。機械語モニタには、機械語プログラム作成のための便利な機能が備わっています。詳細については「第9章　機械語モニタ」を参照してください。

機械語モニタのコマンド

コマンド	機能
A	i8086形式のニーモニックをアセンブル
C	セグメントアドレスの変更
D	メモリの内容を16進数で表示
E	メモリの内容を変更
F	メモリの内容を定数で一括して変更
G	プログラムの実行
I	I/Oポートからデータの読み出し
L	逆アセンブル
M	指定したメモリ領域を別のアドレスに転送
O	I/Oポートへデータを出力
P	プリンタへの出力を制御
S	メモリの内容を変更
X	CPUのレジスタの内容の変更
HELP または CTRL + A	コマンドの種類と形式を画面に表示
CTRL + B	BASICモードへ復帰
CTRL + D	ディスクの内容をセクタ単位で表示
CTRL + P	プリンタへの出力を制御
CTRL + R	ディスクからデータを直接読み出す
CTRL + W	ディスクへデータを直接書き込む

• モニタモードからBASICモードに戻すには CTRL + B キーを押します。

参　照　日本語Disk BASICユーザーズマニュアル　第9章　機械語モニタ

M

NAME

【ネイム】

NAME
名付ける意味から、ファイルの名前を変更すること。

機　能　ディスク上のファイルの名前を変更します。

書　式　NAME　古いファイル名　AS　新しいファイル名

使用例　NAME "OLD" AS "NEW"　←OLD というファイル名をNEWに変更します。

解　説
- **古いファイル名**で指定するファイルの名前を、**新しいファイル名**で指定するファイル名に変更します。
- ファイル名の変更を行うファイルはクローズしてなければなりません。
- **古いファイル名**で指定したドライブ名と**新しいファイル名**で指定するドライブ名は一致していなければなりません。

NEW

【ニュー】

NEW
新しい、新規のという意味からメモリをクリアすること。

機　能　メモリ中のプログラムを消去し新しくプログラムを作成する準備をします。

書　式　NEW

使用例　NEW

解　説
- メモリ中のプログラムを消去するとともに、すべての変数領域を消去(クリア)します。
- NEW コマンド実行後は、BASIC のコマンド入力待ちになります。
- 新しくプログラムを入力する際に、前のプログラムを消去するために使用します。

参　照
CLEAR(変数の消去)
ERASE(配列変数の消去)
DELETE(プログラムの行単位の削除)

NEW ON

【ニュー・オン】

NEW ON
新しくスイッチを ON にする意味。ここではシステムを再起動すること。

機 能　BASIC システムを再起動します。

書 式　NEW ON　機能

使用例　NEW ON 0　←1行40桁、1画面20行の画面モードで起動します。

解 説
- **機能**の値はディップスイッチ SW2 の各ビットに対応した 0 から255の値を指定します。このコマンドを実行すると対応するモードで BASIC システムを再起動します。したがって、このコマンドを実行する前には必要なプログラムやデータは、ディスクなどにセーブしてください。

機能：

意味			ディップスイッチ SW 2
(予約)	常に0にする		1
(予約)	常に0にする		2
1行の桁数	0	40桁/行	3
	1	80桁/行	
画面の桁数	0	20行/画面	4
	1	25行/画面	
未使用	—		5
未使用	—		6
未使用	—		7
未使用	—		8

- ビット 4 からビット 7 までは NEW ON コマンドでは意味を持ちません。

OCT$

【オクト・ダラー】

OCTal
8進数を意味するoctalの省略形。

機　能　10進数を8進表記の文字列に変換します。

書　式　OCT$(数式)

使用例　PRINT OCT$(N)　←Nに相当する10進数を8進文字列に変換し、画面に表示します。

解　説

- 10進数の**数値**を8進表記の文字列に変換します。
- 数値は−32768から65535の範囲です。**数値**が負の数の場合、2の補数形式で表現します。したがって、**数値**を−Nとした場合、OCT$(−N)とOCT$(65536−N)は同じ結果になります。

例

```
PRINT OCT$(-1)
177777
OK
```

```
PRINT OCT$(65535)
177777
OK
```

- 8進表記の数値は、8進数の前に&O(または&o、&)を付けて表現します。またOCT$関数で文字列に変換されたものを10進数の数値に変換するにはVAL("&O"+A$)のようにします。

参　照

HEX$(10進数を16進表記の文字列に変換)
VAL(文字列を数値に変換)

プログラム例

```
100 ' HEX$ / OCT$
110 ' --- 10進数 -> 16進数 -> 8進数 ---
120 PRINT "10進数  16進数  8進数"
130 FOR I=0 TO 16
140   PRINT USING " ###";I;
150   PRINT TAB(12);RIGHT$(" "+HEX$(I),2);
160   PRINT TAB(21);RIGHT$("  "+OCT$(I),3)
170 NEXT
180 END
```

O

ON COM GOSUB

【オン・コム・ゴーサブ】

ON COMmunication GO to SUBroutine
communication は通信回線のこと。全体として通信回線よりデータを受信したらサブルーチンへ行くこと。

機能 通信回線よりデータを受信したら実行する割り込み処理ルーチンの開始行を定義します。

書式 ON COM GOSUB [行番号] [, [行番号]] [, [行番号]]

使用例

```
100 OPEN "COM:" AS #1          ←通信回線をオープンします。
200 ON COM GOSUB 5000          ←割り込みが発生したら5000行に制御を移します。
300 COM ON                     ←割り込みを許可します。
```

解説

- 通信回線よりデータを受信したときに実行する割り込み処理ルーチンの開始行を**行番号**あるいは**ラベル**で指定します。
- 割り込み処理ルーチンから、もとのプログラムに戻るには RETURN 文を使用します。RETURN 文に行番号を指定すると、割り込みが発生した場所以外の行に戻ることができます。
- ON COM GOSUB 文は通信回線をオープンした後に実行します。また、COM ON 文を実行して割り込み処理を許可していないと、割り込み処理ルーチンは実行しません。割り込み処理プログラム作成の流れは次のようになります。

```
100 OPEN "COM:" AS #1
   :
200 ON COM GOSUB 5000
   :
300 COM ON              ↓ 割り込み許可
   :
```

- 割り込み処理ルーチンの実行中は、自動的に割り込み停止の状態になり、RETURN 文でもとのプログラムに戻る際に再び割り込み許可の状態になります。したがって、割り込み処理ルーチンの実行中に、さらに同じ割り込み処理ルーチンを実行することはありません。
- 割り込み処理ルーチンで受信したデータをすべて読み出し、受信バッファ内のデータがなくなっていても RETURN 文により割り込み許可の状態になると再び割り込み処理ルーチンを実行することがあります。このため、データのないときは割り込み処理ルーチンの処理を行わないように処理ルーチンの最初に LOC 関数などを使用してデータの有無を確認してください。

参照

COM ON/OFF/STOP(通信回線からの割り込み制御)
RETURN(サブルーチンからの復帰)

ON ERROR GOTO

【オン・エラー・ゴートゥ】

ON ERROR GOTO
on は～するとという意味で、全体としてエラーになったら～へ行くこと。

機能　エラーが発生したときに、エラー処理を行うエラー処理ルーチンの開始行を定義します。

書式　ON ERROR GOTO　行番号

使用例　ON ERROR GOTO 1000　←エラーが発生したら1000行以下を実行します。

解説
- エラーが発生したときに実行する、エラー処理ルーチンの**開始行**を**行番号**あるいは**ラベル**で指定します。
- エラー処理ルーチンからもとのプログラムに戻るには、RESUME文を使います。
- エラー処理ルーチン内でERR, ERL関数などを使って、エラーの種類を判別してエラー処理を行います。
- エラー処理ルーチン内で再びエラーが発生すると、エラーメッセージを表示してプログラムの実行を終了します。
- この命令実行後に、エラー処理ルーチンの実行を禁止するにはON ERROR GOTO 0(行番号に0を指定)を実行します。実行後にエラーが発生すると、エラーメッセージを表示してプログラムの実行を停止します。
- プログラムの実行の終わりにON ERROR GOTO 0を実行するようにしてください。実行しないとダイレクトモードでのエラーについてもエラー処理ルーチンへ実行が移ります。

O

参照　ERR(エラーコード)
ERL(エラー発生行)
RESUME(エラー処理ルーチンからの復帰)

プログラム例

```
100 ' ON ERROR GOTO
110 ' ---  エラー処理  ---
120 ON ERROR GOTO 200
130 INPUT "A , B";A,B
140 C=A/B
150 PRINT TAB(14);"A/B=";C
160 GOTO 130
170 END
200 ' ---  エラー処理ルーチン  ---
210 PRINT TAB(14);"0で割ることはできません。"
220 RESUME  160
```

```
RUN
A , B? 12,2
              A/B= 6
A , B? 45,0
              0で割ることはできません。
A , B?
```

ON～GOSUB
ON～GOTO

【オン・ゴー・サブ】【オン・ゴー・トゥ】

ON～GO to SUBroutine
～にもとづいてという意味の on から、全体として～にもとづいてサブルーチンへ行くこと。

ON～GO TO
～にもとづいてという意味の on から、全体として～にもとづいて～へ行くこと。

機能 数値に対応させて、実行するサブルーチンや分岐先を変更します。

書式
ON　数式 GOSUB ［行番号］［,［行番号］・・・
ON　数式 GOTO ［行番号］［,［行番号］・・・

使用例 ON A GOTO 1000, 2000, 3000, 4000　←Aが1のとき1000行へ、2のとき2000行へ、3のとき3000行へ、4のとき4000行へ分岐します。

解説
- **数式**の値が1ならば1番目の**行番号**に、2ならば2番目の**行番号**に、というように**数式**の値と**行番号**を対応させます。**数式**の値が負の場合 Illegal function call（違法関数呼び出し）エラーになります。

数式の値	0	1	2	3	…………	255
対応する行番号	次の行	1番目	2番目	3番目	…………	255番目

行番号は、カンマ(,)で区切り必要なだけ並べます。ただし、プログラムの1行の範囲内に限ります。
- サブルーチンを実行するのか、分岐するのかによりGOSUBとGOTOを使い分けます。
- **数値**の値が0の場合または数値の値が行番号の個数よりも大きくなった場合は、その次の文を実行します。
- ON～GOSUB文で実行するサブルーチンから、もとのプログラムに戻るにはRETURN文を使用します。

参照 GOSUB（サブルーチンの呼び出し）
GOTO（プログラムの分岐）

プログラム例
```
100 ' ON - GOSUB / GOTO
110 ' --- メニュー処理の作成 ---
120 PRINT "1. . . 追加"
130 PRINT "2. . . 削除"
140 PRINT "3. . . 一覧"
150 PRINT "4. . . 終了"
160 INPUT "処理の番号を選んでください。";CM
170 IF CM<1 OR CM>4 THEN BEEP : GOTO 160
180 ON CM GOSUB 200,210,220,230
190 END
200 PRINT "追加" : RETURN
210 PRINT "削除" : RETURN
220 PRINT "一覧" : RETURN
230 PRINT "終了" : RETURN
```

ON HELP GOSUB

【オン・ヘルプ・ゴーサブ】

ON HELP key GO to SUBroutine
on HELP key で HELP キーを押したらという意味。全体として HELP キーを押したらサブルーチンを実行すること。

機能 HELP キーを押すと実行する割り込み処理ルーチンの開始行を定義します。

書式 ON HELP GOSUB　行番号

使用例

100 ON HELP GOSUB 5000	←割り込みが発生したら5000行へ制御を移します。
120 HELP ON	←割り込みを許可します。

解説

• HELP キーを押すと実行する割り込み処理ルーチンの開始行を**行番号**あるいは**ラベル**で定義します。
• 割り込み処理ルーチンから、もとのプログラムに戻るには RETURN 文を使用します。RETURN 文に行番号を指定すると、割り込みが発生した場所以外の行に戻ることができます。
• HELP ON 文を実行して割り込み処理を許可していないと割り込み処理ルーチンは実行しません。割り込み処理プログラム作成の流れは、次のようになります。

```
100 ON HELP GOSUB 5000
  ⋮
120 HELP ON          ↓ 割り込み許可
  ⋮
```

• 割り込み処理ルーチンの実行中は、自動的に割り込み停止の状態になり、RETURN文でもとのプログラムに戻る際に再び割り込み許可の状態になります。したがって、割り込み処理ルーチンの実行中に、さらに同じ割り込み処理ルーチンを実行することはありません。

参照 HELP ON/OFF/STOP(HELP キーによる割り込みの制御)
RETURN(サブルーチンからの復帰)

O

ON KEY GOSUB

【オン・キー・ゴーサブ】

ON function KEY GO to SUBroutine
on key でファンクションキーを押したらという意味。全体としてファンクションキーを押したらサブルーチンへ行くこと。

機　能　ファンクションキーを押すと実行する割り込みの処理ルーチンの開始行を定義します。

書　式　ON KEY GOSUB［行番号］［,［行番号］・・・］

使用例

100 ON KEY GOSUB 5000　←割り込みが発生したら5000行へ制御を移します。

120 KEY(1)　ON　←割り込みを許可します。

解　説

- ファンクションキーを押すと実行する割り込み処理ルーチンの開始行を**行番号**あるいは**ラベル**で定義します。**行番号**はカンマ(,)で区切って最大10個まで並べることができます。この並びの順序はファンクションキーの番号と一対一に対応します。
- 割り込み処理ルーチンからもとのプログラムに戻るにはRETURN文を使用します。RETURN文に行番号を指定すると、割り込みが発生した場所以外の行に戻ることができます。
- KEY ON文を実行して割り込み処理を許可していないと割り込み処理ルーチンは実行しません。割り込み処理プログラム作成の流れは次のようになります。

```
100 ON KEY GOSUB 5000
 :
120 KEY(1) ON        割り込み許可
 :
```

- 割り込み処理ルーチンの実行中は、自動的に割り込み停止の状態になり、RETURN文でもとのプログラムに戻る際に再び割り込み許可の状態になります。したがって、割り込み処理ルーチンの実行中に、さらに同じ割り込み処理ルーチンを実行することはありません。

参　照

KEY ON/OFF/STOP(ファンクションキーによる割り込み制御)
RETURN(サブルーチンからの復帰)

プログラム例

```
100 ' ON KEY GOSUB
110 ' ---  ﾌｧﾝｸｼｮﾝｷｰ・3による割り込み  ---
130 ON KEY GOSUB ,,200
140 KEY(3) ON
150 PRINT CT;
160 GOTO 150
170 '
200 BEEP
210 CT=CT+1:IF CT=10 THEN CT=0
220 RETURN 150
```

ON PEN GOSUB

【オン・ペン・ゴーサブ】

ON lightPEN GO to SUBroutine
on pen でライトペンを押したらという意味。全体としてライトペンを押したらサブルーチンへ行くこと。

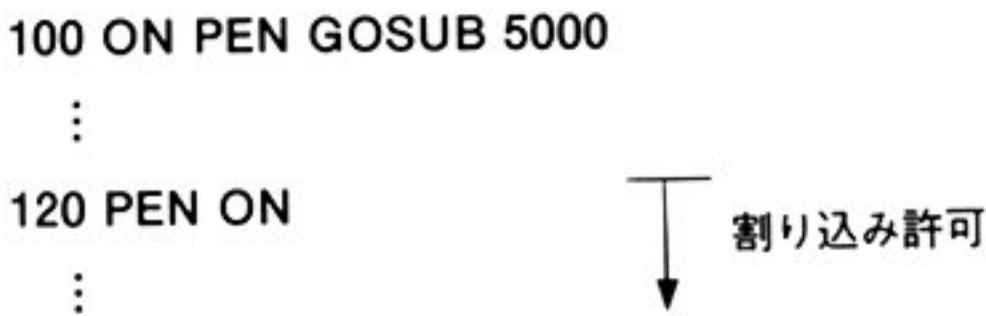
機能　ライトペン入力によって実行する割り込み処理ルーチンの開始行を定義します。

書式　ON PEN GOSUB　行番号

使用例

```
100 ON PEN GOSUB 5000    ←割り込みが発生したら5000行以下を実行します。
120 PEN ON               ←割り込みを許可します。
```

解説

- ライトペン入力によって実行する割り込み処理ルーチンの開始行を**行番号**あるいは**ラベル**で指定します。
- 割り込み処理ルーチンからもとのプログラムに戻るには RETURN 文を実行します。RETURN 文に行番号を指定すれば、割り込みが発生した場所以外の行に戻ることができます。
- PEN ON 文を実行して割り込み処理を許可していないと割り込み処理ルーチンは実行しません。割り込み処理プログラム作成の流れは次のようになります。

```
100 ON PEN GOSUB 5000
  :
120 PEN ON           ↓ 割り込み許可
  :
```

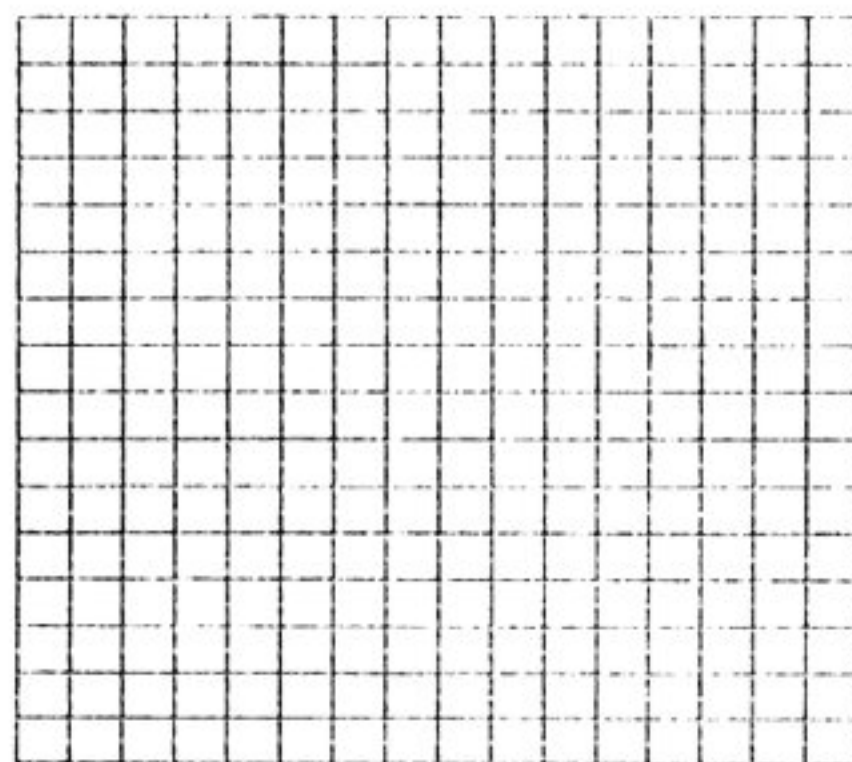
- 割り込み処理ルーチンの実行中は、自動的に割り込み停止の状態になり、RETURN 文でもとのプログラムに戻る際に再び割り込み許可の状態になります。したがって、割り込み処理ルーチンの実行中に、さらに同じ割り込み処理ルーチンを実行することはありません。

参照　PEN ON/OFF/STOP（ライトペンによる割り込み制御）
RETURN（サブルーチンからの復帰）

プログラム例

- このプログラムは漢字コード&H7621にユーザー定義文字を設定するものです。210行を変更することによりほかのコードに設定することもできます。
- プログラムを実行すると16×16の□が画面に表示されますので、これにライトペンを押しつけてください。
- 一回押しつけるとドットの表示(青で表示)、もう一回押すとドットの消去(白で表示)します。
- この画面で作成した文字は画面の左中のワク内に表示されます。
- 終了する場合は終了とかかれている□をライトペンで押します。

```
100 ' ON PEN GOSUB
110 ' PEN ON/OFF/STOP
120 ' PEN
130 ' --- ライトペンで外字を登録する ---
140 '
150 WIDTH 80,25 : CONSOLE 0,25,0,1
160 SCREEN 3 : COLOR ,,,,0
170 WINDOW(0,0)-(639,399) : VIEW(0,0)-(639,399)
180 CLS 3
190 '
200 DIM KP%(17) : KP%(0)=16 : KP%(1)=16
210 KC=&H7621
220 KPLOAD KC,KP%
230 '
240 GOSUB *KPCLS
250 '
260 LOCATE 38,3 : COLOR 7 : PRINT "■■ : 消  去"
270 LOCATE 38,5 : COLOR 7 : PRINT "■■ : 終  了"
280 '
290 ON PEN GOSUB 490 : PEN ON
300 LOCATE 38,9 : PRINT "JIS漢字コード : ";HEX$(KC)
310 LINE (444,172)-STEP(22,22),3,B
320 LOCATE 56,11 : COLOR 7
330 PRINT KNJ$("1B4B")+KNJ$(HEX$(KC))+KNJ$("1B48")
340 '
350 *PENLOOP
360   KPLOAD KC,KP%
370   GOTO *PENLOOP
380 END
390 '
400 *KPCLS
410   FOR I=2 TO 17 : KP%(I)=0 : NEXT
420   COLOR 7
430   FOR I=0 TO 31
440     FOR J=0 TO 15
450       LOCATE I,J : PRINT "■";
460     NEXT
470      NEXT
480   RETURN
490 '
500 '  ライトペン割り込み処理
510 '
520 PX=PEN(1) : PY=PEN(2) : PEN OFF
530   IF 37<PX AND PX<41 THEN *SELECT
540   IF PX>31 OR PY>31 THEN *EXIT
550 '
560   PX=FIX(PX/2)
570   IF PX>7 THEN BIT=23-PX ELSE BIT=7-PX
580   BPT=VAL("&H"+HEX$(2^BIT))
590   CD=KP%(PY+2) AND BPT
600   IF CD=0 THEN *KPSET ELSE *KPRESET
610 '
620 *SELECT
630   IF PY=3 THEN GOSUB *KPCLS ELSE *KPEND
640   GOTO *EXIT
650 *KPSET
660   COLOR 1 : LOCATE PX*2,PY : PRINT "■■";
670   KP%(PY+2)=KP%(PY+2) OR BPT
680   BEEP : GOTO *EXIT
```

```
690 *KPRESET
700   COLOR 7 : LOCATE PX*2,PY : PRINT "■";
710   KP%(PY+2)=KP%(PY+2) AND (NOT BPT)
720   BEEP : GOTO *EXIT
730 *EXIT
740   FOR I=1 TO 200 : NEXT
750   PEN ON : KPLOAD KC,KP% : RETURN  *PENLOOP
760 '
770 *KPEND
780  COLOR 7 : CLS 3 : PEN OFF : END
```

■ : 消　去

■ : 終　了

JIS漢字コード : 7621

O

ON STOP GOSUB

【オン・ストップ・ゴーサブ】

ON STOP KEY GO to SUBroutine
on STOP で STOP キーを押したらという意味。全体として STOP キーを押したらサブルーチンを実行すること。

機能　[STOP] キーを押すと実行する割り込み処理ルーチンの開始行を定義します。

書式　ON STOP GOSUB　行番号

使用例

100 ON STOP GOSUB 5000	←割り込みが発生したら5000行以下を実行します。
120 STOP ON	←割り込みを許可します。

解説

- [STOP] キーあるいは [CTRL] ＋ [C] キーを押すと実行する割り込み処理ルーチンの開始行を**行番号**あるいは**ラベル**で定義します。
- 割り込み処理ルーチンからもとのプログラムに戻るには RETURN 文を実行します。RETURN 文に行番号を指定すれば、割り込みが発生した場所以外の行に戻ることができます。
- STOP ON 文を実行して割り込み処理を許可していないと割り込み処理ルーチンは実行しません。割り込み処理プログラム作成の流れは次のようになります。

```
100 ON STOP GOSUB 5000
  :
120 STOP ON          ┬ 割り込み許可
  :                  ↓
```

- 割り込み処理ルーチンの実行中は、自動的に割り込み停止の状態になり、RETURN 文でもとのプログラムに戻る際に再び割り込み許可の状態になります。したがって、割り込み処理ルーチンの実行中に、さらに同じ割り込み処理ルーチンを実行することはありません。
- 不用意にこの命令を実行すると、[STOP] キーおよび [CTRL] ＋ [C] キー本来の機能を実行することができなくなり、プログラムの実行を停止することができなくなります。

参照

RETURN(サブルーチンからの復帰)
STOP ON/OFF/STOP(STOP キーによる割り込みの制御)

ON TIME$ GOSUB

【オン・タイム・ダラー・ゴーサブ】

ON TIME GO to SUBroutine
on time は時刻の意味。全体として指定した時刻になったらサブルーチンを実行すること。

機能　指定した時刻に実行する割り込み処理ルーチンの開始行を定義します。

書式　ON TIME$="時：分：秒" GOSUB　行番号

使用例

```
100 ON TIME$="00：10：00" GOSUB 5000   ←割り込みが発生したら5000行以下を実行します。
110 TIME$="00：00：00"
120 TIME$ ON                           ←割り込みを許可します。
```

解説

- 指定した時刻**"時：分：秒"**に実行する割り込み処理ルーチンの開始行を**行番号**あるいは**ラベル**で定義します。**"時：分：秒"**の指定は TIME$ 文と同じ形式です。
- ON TIME$ GOSUB 文で制御できる最小単位は 2 秒です。したがって、指定した時間より ± 1 秒程度の誤差を生じることがあります。
- 割り込み処理ルーチンからもとのプログラムに戻るには RETURN 文を実行します。RETURN 文に行番号を指定すれば、割り込みが発生した場所以外の行に戻ることができます。
- TIME$ ON 文を実行して割り込み処理を許可していないと割り込み処理ルーチンは実行しません。割り込み処理プログラム作成の流れは次のようになります。

```
100 ON TIME$="00：10：00" GOSUB 5000
  ：
120 TIME$ ON                       ↓ 割り込み許可
  ：
```

- 割り込み処理ルーチンの実行中は、自動的に割り込み停止の状態になり、RETURN 文でもとのプログラムに戻る際に再び割り込み許可の状態になります。したがって、割り込み処理ルーチンの実行中に、さらに同じ割り込み処理ルーチンを実行することはありません。

参照

RETURN（サブルーチンからの復帰）
TIME$（時間の設定）
TIME$ ON/OFF/STOP（時間による割り込みの制御）

O

OPEN

【オープン】

OPEN
開くことから、ファイルを開いて読み書きできる状態にすること。

機　能　ファイルに対してデータの入出力ができるように準備をします。これをファイルをオープンするといいます。

書　式　OPEN　ファイル指定子［FOR {INPUT | OUTPUT | APPEND}］AS［#］ファイル番号

使用例

```
OPEN "TEST" FOR OUTPUT AS #1
```

←TESTというファイルを出力モードでオープンします。

解　説

• **ファイル指定子**で指定したファイルを、指定した**ファイル番号**でオープンします。**ファイル指定子**で指定できるデバイス名は次のとおりです。ディスクの場合はファイル名を指定する必要があります。また通信回線の場合はファイル名の代わりに通信パラメータを指定します。

デバイス名	意　　味
KYBD:	キーボード
SCRN:	画面
LPT1:	プリンタ
1:	ディスク1
2:	ディスク2
3:	ディスク3
⋮	⋮
10:	ディスク10
COM1:	通信回線1（本体内蔵）
COM2:	通信回線2
COM3:	通信回線3

• FOR以下にファイルへの入出力方法を指定します。デバイスによっては指定できないものがありますので注意してください。

機　　能	入出力方法
省略	ランダムファイル　　　：レコード番号を指定して入出力
INPUT	シーケンシャルファイル：ファイルの先頭から順に入力
OUTPUT	シーケンシャルファイル：ファイルの先頭から順に出力
APPEND	シーケンシャルファイル：ファイルの終わりから順に出力

・ファイル番号は1から15の範囲の整数を指定します。ただし、BASIC起動時に入力したファイル数(How many files?に答えた数)以下でなければなりません。ファイル番号はオープンしてからクローズするまでの間、そのファイルを参照するために用いられる番号で、入出力命令は、この番号を使ってファイルを区別します。内部的にはファイルをオープンするとファイル番号に対応した入出力用バッファが割り当てられ、ファイルをクローズすると、この領域が開放されます。したがってファイルをクローズするまでは、同じファイル番号を別のファイルに指定することはできません。

・通信回線の通信パラメータ

通信回線をオープンするには、次のようにデバイス名に続けて通信パラメータを指定します。

"COM[回線番号]：(pcsxh)"

p：パリティチェック	E：偶数パリテイ O：奇数パリティ N：パリティなし
c：データビット	7：7ビット 8：8ビット
s：ストップビット	1：1ビット 2：1.5ビット 3：2ビット
x：XON/XOFF制御	X：XON/XOFF制御を行う N：XON/XOFF制御を行わない
h：SI/SO制御	S：SI/SO制御を行う N：SI/SO制御を行わない

省略した場合はメモリスイッチの値を指定したことになります。通信速度の設定はメモリスイッチで行います。

・指定できる入出力モードは次のとおりです。

デバイス名	INPUT	OUTPUT	APPEND	省略
KYBD：	○	×	×	○
SCRN：	×	○	×	○
LPT1：	×	○	×	○
1：～10：	○	○	○	○
COM1：～COM3：	○	○	○	○

OPTION BASE

【オプション・ベイス】

OPTION BASE
option は選択の意味。ここでは配列の基底(ベイス)を選択すること。

機能　配列の添字の下限(基底)を0または1に設定します。

書式　OPTION BASE |0|
　　　　　　　　　　 |1|

使用例

```
OPTION BASE 1
```
←配列の添字を下限を1にします。

解説

- **配列の添字の最小値**を0または1にします。
- BAISC 起動時は0になっています。OPTION BASE 1を実行し、下限を1にした後、配列の添字として0を用いると Subscript out of range(添字が範囲外にある)エラーになります。
- この宣言は、プログラム中で配列変数を宣言、あるいは使用した後に行うことはできません。またCHAIN文によって配列変数が引き渡された側のプログラムで宣言することはできません。このような場合には、Duplicate Definition(二重定義)エラーになります。
- 添字の下限は、RUNあるいはCLEAR文を実行した後、変更することができます。

参照　DIM(配列変数の宣言)
日本語 Disk BASIC ユーザーズマニュアル　第2章　BASICの文法

OUT

【アウト】

OUT
外に、外へという意味から、データを I/O ポートに送り出すこと。

機　能　I/O ポートへ直接 1 バイトのデータを送ります。

書　式　OUT　ポート番号，出力データ

使用例　OUT &H68, &H0D　　←ポート &H68 にデータを出力します。

解　説

- **ポート番号**で指定した I/O ポートに 1 バイトの**出力データ**を出力します。
- **ポート番号**は－32768から65535までの数値を指定します。負の数を指定した場合は65536に指定した値を加算した結果を指定したことになります。

参　照　INP(I/Oポートから直接入力)

O

PAINT

【ペイント】

PAINT
色を塗ること。画面に色を付けること。

機　能　画面の指定範囲を色や模様で塗りつぶします。

書　式　PAINT（Wx, Wy）[, [領域色 | タイルストリング] [, [境界色]]]

使用例

PAINT（100, 100）, 6 , 7　←座標(100, 100)が内側にあるパレット番号7の色で囲まれた領域を、パレット番号6の色で塗りつぶします。

PAINT（100, 100）, TIL$, 7　←上記と同じ領域を指定した模様(タイルパターン)で塗りつぶします。

解　説

- ワールド座標(**Wx, Wy**)を含んだ**境界色**で囲まれた領域を、指定した**領域色**や、模様(**タイルストリング**)で塗りつぶします。
- ワールド座標(**Wx, Wy**)は塗り始める座標です。指定する座標はビューポート内になければいけません。そうでない場合はIllegal function call(違法関数呼び出し)エラーになります。相対座標で指定することもできます。
- **領域色**には、塗りつぶす色をパレット番号で指定します。**領域色**の代わりに模様で塗りつぶす場合は、**タイルストリング**を指定します。
- **領域色**、**タイルストリング**とも省略した場合はCOLOR文で設定した前景色を指定したことになります。
- **境界色**には、塗りつぶす領域の境界をパレット番号で指定します。(**Wx, Wy**)の座標で指定した位置から、ここで指定した色までの領域を塗りつぶします。省略した場合は、領域色で指定した色を指定したことになります。
- 指定した座標(**Wx, Wy**)の点がすでに境界色と同じ色である場合はPAINT文は何の動作もしません。
- すでに**タイルストリング**で塗りつぶした領域に別の模様を塗る場合は、多少時間がかかることがあります。また内部のスタックを多量に使うためOut of memory(メモリの不足)エラーになることがあります。この場合はCLEAR文でスタックの大きさを大きくしてください。
- STOPキーによりPAINT文の実行を中断することができます。

・PAINT 文は、ビューポート内のみで動作します。したがって、ビューポートの境界が、PAINT 文の動作の境界になります。

・**タイルストリング**の構成は次のとおりです。

タイルストリングは模様の大きさと色を指定する文字列です。タイルストリングの形式は横方向8ドットを1単位とし、縦方向に必要な単位数分を定義したものです。縦方向をnドットとした場合の必要な文字列の長さは次のとおりです。

白黒モード　：　nバイト

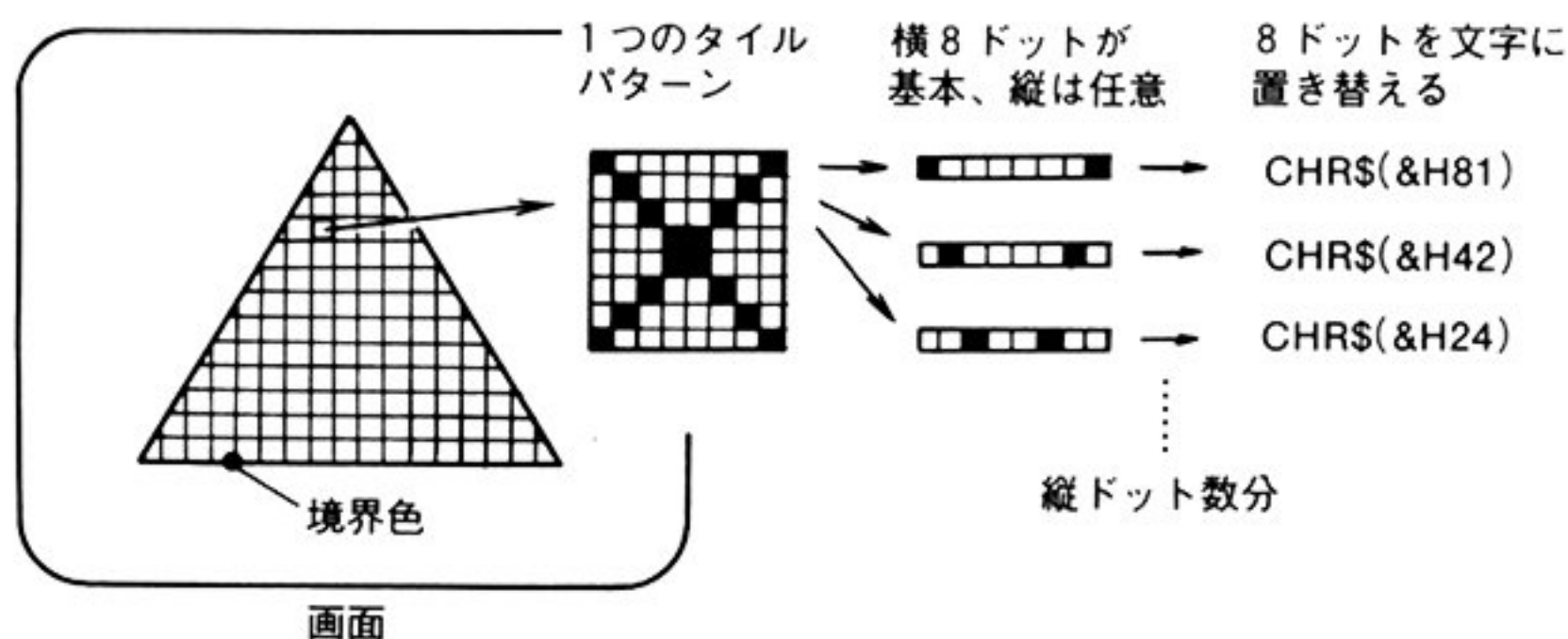
カラーモード(4096色中16色モード)　：　4×nバイト

(4096色中8色モード)　：　3×nバイト

(8色中8色モード)　：　3×nバイト

(1) 白黒モード

・画面上の横8ドットが1バイト(8ビット)の各ビットに対応します。これを、CHR$関数などを用いて、文字列に変換します。

・模様の縦方向のドット数分に上記の変換を行い、上から順に並べた文字列がタイルストリングになります。

タイルストリング＝CHR$(&H81)＋CHR$(&H42)＋CHR$(&H24)＋……

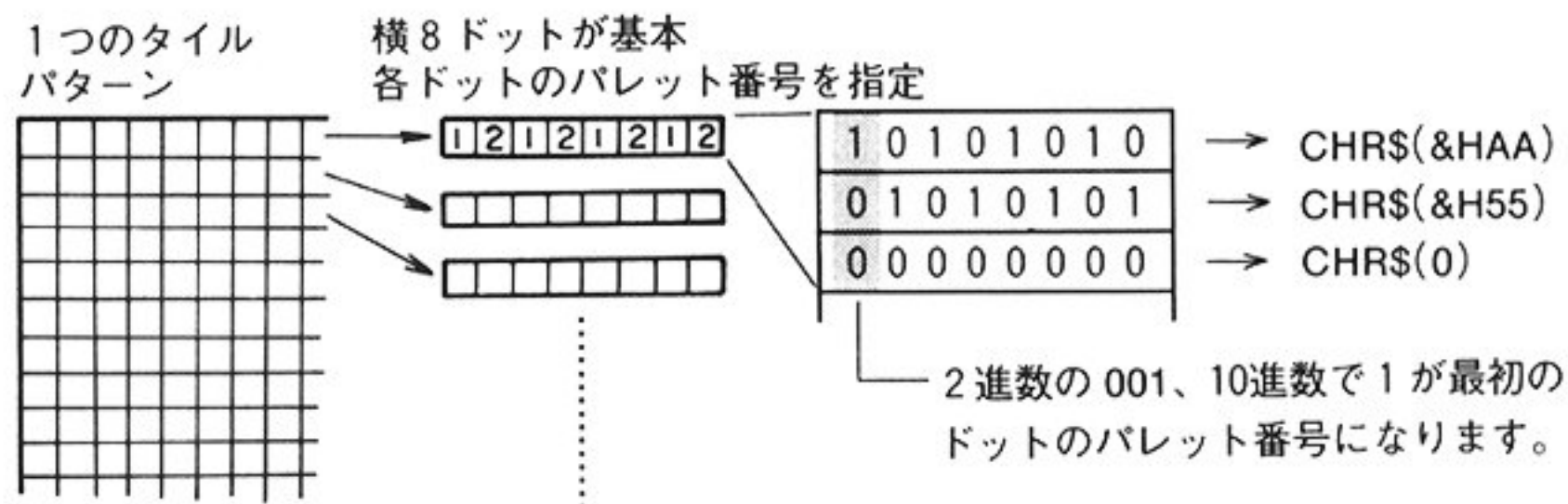
(2) カラーモード

・画面上の横8ドットが1バイトの各ビットに対応することは白黒モードの場合と同じです。ただし色を指定するために1ドット当たり3ビット(8色モード)、または4ビット(16色モード)必要になります。

・したがって、横8ドットを表示するのに3バイトあるいは4バイト必要になります。これをドットごとに対応させ、表示する色をパレット番号で指定します。

・文字列の長さは3あるいは4の倍数になります。文字列の長さに余りがある場合は無視しますが、3バイトあるいは4バイトに満たない場合は Illegal function call(違法関数呼び出し)エラーになります。

P

8色モード

16色モード

横8ドットを表現するのに4文字分必要になります。

• PAINT 文実行後、最終参照座標は(Wx, Wy)に移動します。

参照　COLOR(パレットの設定)

プログラム例

```
100 ' PAINT
110 '----  領域をタイルパターンで埋める  ----
120 '
130 SCREEN 3,0 : COLOR : CLS 3
140 CIRCLE(100,100),100,7
150 FOR I=1 TO 24
160   READ A$ : TILE$=TILE$+CHR$(VAL("&H"+A$))
170 NEXT
180 PAINT(100,100),TILE$,7
190 END
200 DATA 66,99,7E,BD,42,BD,DB,24,DB,66,99,E7
210 DATA 66,99,E7,DB,24,DB,BD,BD,42,66,99,7E
```

PEEK

【ピーク】

PEEK
のぞくという意味。

機能　指定したメモリから1バイトのデータを読み出します。

書式　PEEK(アドレス)

使用例　A＝PEEK(&HF000)　←&HF000の内容をAに代入します。

解説
- アドレスで指定したメモリから1バイトのデータを読み出します。指定するアドレスはDEF SEG文で設定したセグメントアドレスのオフセットアドレスです。

参照　DEF SEG(セグメントアドレスの設定)
POKE(メモリへのデータ書き込み)

P

PEN

【ペン】

PEN
ペンはここではライトペンの意味。

機能　ライトペンの状態を調べます。

書式　PEN(機能)

使用例　X=PEN(1)　←ライトペンを押している位置のキャラクタ座標上のX座標を変数Xに代入します。

解説
- 機能で指定したライトペンの情報を返します。機能には0から2までの数値を指定します。

機能	意味
0	ライトペンを押しているときに－1(真)を返し、離しているときに0(偽)を返します。真になったときはON PEN GOSUB文で定義した割り込み処理ルーチンに入るまで－1(真)を保持し、ルーチンに入ったときに0(偽)になります。
1	ライトペンを押しているとき、その場所をキャラクタ座標のX座標値で返します。
2	ライトペンを押しているとき、その場所をキャラクタ座標のY座標値で返します。

- PEN関数は、PEN ON文を実行していないと有効になりません。

参照
ON PEN GOSUB(ライトペンによる割り込み処理ルーチンの定義)
PEN ON/OFF/STOP(ライトペンによる割り込み制御)

PEN ON
PEN OFF
PEN STOP

【ペン・オン】
【ペン・オフ】
【ペン・ストップ】

PEN
ペンはここではライトペンの意味。ライトペンに関する制御を行うこと。

機能　ライトペンの割り込みを可能な状態、禁止の状態、停止の状態にします。

書式

```
PEN |ON  |
    |OFF |
    |STOP|
```

使用例

PEN ON	←割り込みを可能な状態に設定します。
PEN OFF	←割り込みを禁止の状態に設定します。
PEN STOP	←割り込みを停止の状態に設定します。

解説

- **PEN ON** はライトペンからの割り込みを可能な状態にします。この命令を実行後は、ライトペンを押して光を感じると ON PEN GOSUB 文で定義した割り込み処理ルーチンを実行します。
- **PEN OFF** はライトペンからの割り込みを禁止します。この命令を実行後は、ライトペンを押しても割り込み処理を実行しません。
- **PEN STOP** はライトペンによる割り込みを停止します。この命令を実行後は、ライトペンを押しても割り込み処理を実行しません。ただし、データの記憶は行いますので、この後、PEN ON を実行すると、ただちに ON PEN GOSUB 文で指定した割り込み処理ルーチンを実行します。
- プログラムの実行を終了するときは、PEN OFF を実行してください。
- ON PEN GOSUB 文で定義した割り込み処理ルーチンでは、自動的に割り込み停止状態にします。また PEN 関数の値を 0 (偽)にします。

参照　ON PEN GOSUB(ライトペンによる割り込み処理ルーチンの定義)
PEN(ライトペンの情報)

POINT

【ポイント】

POINT
点、地点という意味。ここでは最終参照座標のこと。

機　能　最終参照座標を移動します。

書　式　POINT(Wx, Wy)

使用例　POINT(50, 50)　　←最終参照座標を(50, 50)に変更します。

解　説

- 最終参照座標をワールド座標(Wx, Wy)に移動します。座標の前にSTEPを付けて相対座標で指定することもできます。
- 最終参照座標はグラフィック命令を実行すると変化するワールド座標系の座標で、相対座標はこの座標からの相対的な位置を示します。

参　照　日本語Disk BASICユーザーズマニュアル　第6章　テキスト画面とグラフィック画面

POINT[1]

【ポイント】

POINT
点、地点という意味。ここでは最終参照座標のこと。

機能　最終参照座標の位置を与えます。

書式　POINT(機能)

使用例　LX=POINT(2)　←最終参照座標のX座標をスクリーン座標系のX座標としてLXに代入します。

解説

- **機能**に対応した最終参照座標の値を返します。**機能**は0から3までの数値です。

機能	意　味
0	最終参照座標のX座標をワールド座標系のX座標で返します。
1	最終参照座標のY座標をワールド座標系のY座標で返します。
2	最終参照座標のX座標をスクリーン座標系のX座標で返します。
3	最終参照座標のY座標をスクリーン座標系のY座標で返します。

- 最終参照座標がウィンドウの外にあった場合は、返す値は画面上に表示しない点になります。また、このときの値がスクリーン座標の値の場合には、この座標に対応する点はビューポート外になります。

参照　POINT(最終参照座標の変更)

P

POINT^2

【ポイント】

POINT
点、地点という意味。ここでは指定した点の色を知らせること。

機能　指定した座標の点の色を返します。

書式　POINT(Sx, Sy)

使用例　C=POINT(100, 100)　←スクリーン座標(100,100)の点の色(パレット番号)をCに代入します。

解説
- スクリーン座標(Sx, Sy)の点の色をパレット番号で返します。
- ビューポートの外の座標を指定した場合は、－1になります。
- 画面が白黒モードの場合には、白(セット)のとき1、黒(リセット)のとき0を返します。

参照　COLOR(パレットの設定)

プログラム例

```
100 ' POINT2
110 '----  ドットの色を調べる  ----
120 SCREEN 0,0:CLS 3
130 FOR I=0 TO 5
140   C=INT(RND*7+1)
150   LINE(0,I*20+5)-(639,I*20+10),C,B
160   LOCATE 0,I*1+15:PRINT "COLOR :";POINT(POINT(2),POINT(3))
170 NEXT I
180 END
```

POKE

【ポーク】
POKE
突くという意味。ここではメモリにデータを書き込むこと。

機能　指定したメモリに1バイトのデータを書き込みます。

書式　POKE　アドレス，データ

使用例　POKE &H100，&H41　←&H100に&H41("A")というデータを書き込みます。

解説
- **アドレス**で指定したメモリに1バイトの**データ**を書き込みます。この**アドレス**はDEF SEG文で設定したセグメントアドレスのオフセットアドレスになります。
- POKE文は直接メモリの内容を書き換えます。使用する場合は、メモリマップなどで使用可能な領域かどうかを確認してください。誤ってBASICの作業領域を破壊しないように注意してください。

参照
DEF SEG(セグメントアドレスの設定)
PEEK(メモリからデータ読み込み)

P

POS

【ポス/ポジション】

POSition
位置のこと。ここではカーソルの桁位置。

機能　テキスト画面上のカーソルの桁位置を返します。

書式　POS(ダミー)

使用例　A＝POS(0)　←カーソルの桁位置をAに代入します。

解説
- カーソルの現在の桁位置をキャラクタ座標のX座標で示します。
- (　)内は**ダミー**であり、何を指定しても構いません。通常は0を用います。
- カーソルの行位置(キャラクタ座標のY座標)にはCSRLIN関数を使用します。

参照　CSRLIN(カーソルの行位置)

PRINT
LPRINT

【プリント】【エル・プリント】

PRINT
印刷する意味から、ここでは画面に表示すること。

Line PRINT
line はここではプリンタの意味。したがってプリンタに出力すること。

機能 画面やプリンタにデータを出力します。

使用例

PRINT"2*2=";2*2 ←画面に2*2=4と表示します。
LPRINT"ABC" ←プリンタにABCと印字します。

解説

- PRINT 文は、**式**に指定した数値や文字列データを画面に表示します。LPRINT 文はこれらをプリンタに出力します。
- **式**を省略すると復帰/改行だけを行います。
- 2つ以上の**式**を指定するときは、**式**と**式**の間をカンマ(,)、セミコロン(;)、空白のいずれかで区切って並べます。この区切り記号によって表示結果や、印字結果が異なってきます。

①カンマ(,)の場合

BASIC は各行にあらかじめ左から14桁ごとに表示(印字)開始位置を持っています。式と式の間をカンマで区切った場合は、それぞれのデータは同じ行の右側にある最も近い表示開始位置から順次表示(印字)していきます。

```
PRINT"ABC", "DEF", "GHI"
LPRINT"ABC", "DEF", "GHI"
```

P

次の表示開始位置がその行に存在しない場合は、次の行の表示開始位置に出力します。また一番最後の式の終わりがカンマで終わっている場合は、次に実行するPRINT文やLPRINT文のデータは次の表示開始位置から出力します。

②セミコロン(;)の場合

次のデータを詰めて出力します。また一番最後の式の終わりがセミコロンで終わっている場合は、次に実行するPRINT文やLPRINT文のデータは出力したデータに続けて表示(印字)します。

③空白の場合

セミコロンの場合と同じ結果になります。

- 最後の式の後ろに、カンマまたはセミコロンを指定すると、次に出力するデータは前のデータに続けて同じ行に出力します。最後の式の後ろに何も付けないと自動的に復帰/改行を行います。
- 数値データを出力すると、その後ろに1桁の空白が付けられます。また数値の前には、正の数であれば空白を、負の数であればマイナス(-)を出力します。
- 単精度数値の場合、7桁以下の数字で精度に影響を与えずに表示できる数値は、固定小数形式で表示します。7桁の固定小数形式で表示できないものは指数形式で表示します。
- 倍精度数値の場合、16桁以下の数字で精度に影響を与えずに表示できる数値は、固定小数形式で表示します。16桁の固定小数形式で表示できないものは指数形式で表示します。
- PRINT文は省略形として疑問符(?)で代用することができます。
- PRINT文やLPRINT文の中の任意の位置に1回だけ **USING** を記述することができます。このとき

は、次の記述と同等です。

PRINT [式 | , ; 空白 |] ・・・; : PRINT USING 書式文字列 ; [式 | , ; |] ・・・

参照　PRINT USING/LPRINT USING(書式設定)

PRINT USING
LPRINT USING

【プリント・ユージング】
【エル・プリント・ユージング】

PRINT USING
～を用いて表示するという意味から、書式指定をして表示する。

Line PRINT USING
書式指定をしてプリンタ(line)に印字する。

機能 数値や文字列を、書式を指定して画面に表示したりプリンタに印字します。

書式

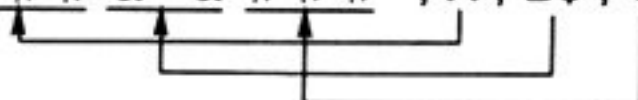
```
PRINT USING  書式文字列；式 |,|  ・・・
                            |;|
LPRINT USING 書式文字列；式 |,|  ・・・
                            |;|
```

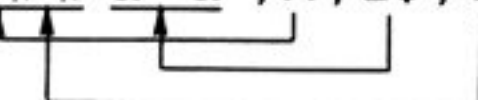
使用例

```
PRINT USING "&      &####.##";A$, B   ←Bを小数点以下2桁の数値で表示します。
LPRINT USING "&      &####.##";A$, B  ←同じようにプリンタに印字します。
```

解説

- PRINT USING文は指定された**書式文字列**にしたがって**式**の値を表示します。LPRINT USING文は、同様にしてプリンタに出力します。
- **書式文字列**は書式を指定するもので、そのままの形では文字として出力しない#、&&、!、@などの書式制御文字と、そのまま画面に出力する文字から構成します。
- **式**と**式**の間はカンマあるいセミコロンで区切っても違いはありません。PRINT文とは異なります。
- **書式制御文字**と**式**は、それぞれ左から順に一対一に対応します。

例 PRINT USING"###& &###";A;B$;C

- **式**の数が、**書式制御文字**の数より多い場合は、最初の**書式指定**より繰り返し対応します。

例 PRINT USING"###& &";A;B$;C

- **式**の数が、**書式制御文字**より少ない場合は、余った書式指定部分は無視します。

例 PRINT USING"###& &###";A

- **書式制御文字**には数値と文字に対応した2つの種類があります。**書式制御文字**の型と式の型が一致していない場合はType mismatch(型の不一致)エラーになります。
- 漢字などの2バイト文字に対して書式制御を行うことはできません。

P

書式制御文字

① **数値の場合**

#	表示する数値の桁数を指定します。ピリオド(.)で小数点位置を指定しない場合は、数値の整数部分を右詰めで表示します。余った桁は空白になります。
.	#で指定する桁数指定の間に挿入し、小数点位置を指定します。小数点以下の数値は小数点の右側に、指定した#の個数に合わせて四捨五入して表示します。また小数点以下の部分で冗長となる桁には0を表示します。
,	桁数指定#の中に置き、数値の整数部を3桁ごとにカンマ(,)で区切って表示します。小数点位置を示すピリオド(.)より左側に置かなければ機能しません。
+	書式指定の先頭または最後に付け、表示する数値の先頭あるいは最後に符号(+あるいは−)を表示します。
−	書式の最後に付け、表示する数値が負の場合に数値の後ろに負符号(−)を表示します。
＊＊	書式文字列の先頭に付けます。数値の左側の桁に空白ができたときに、その部分をアスタリスク(＊)で埋めて表示します。この"＊＊"は同時に2桁分の#を意味します。
¥¥	書式文字列の先頭に付けます。数値の前に円記号(¥)を1つ表示します。この¥¥は同時に2桁分の#を意味しますが、1桁分は¥の表示桁になります。
＊＊¥	書式文字列の先頭に付けます。上記＊＊と¥¥の両方の機能を持ちます。この＊＊¥は同時に3桁分の#を意味しますが、1桁分は¥の表示桁になります。
^^^^	桁数指定#の後ろに付けます。数値を指数形式で表示します。

• 指定した桁数より数値データが大きい場合は、数値の前に"%"を表示し、桁数が足りないことを示します。

② 文字列の場合

!	指定した文字列の最初の1文字(1バイト)を表示します。
&　　　& 空白	指定した文字列の最初から空白の個数+2文字分の文字を表示します。指定した文字列の長さが、これより短い場合は右側に空白を補って表示し、長い場合は、切り捨てて表示します。
@	指定した文字列をそのまま表示します。
_	このアンダーバーに続く1文字をそのまま表示します。書式制御文字として使用している文字をそのまま表示することができます。

③ 制御文字以外

前述の書式制御文字以外の文字を指定した場合は、その文字をそのまま表示します。漢字などの2バイト文字を指定することもできます。

プログラム例

```
100 ' PRINT USING
110 '----  書式を指定してデータを画面に表示する  ----
120 '
130 PRINT USING "!";"EPSON WORLD"
140 PRINT USING "&   &";"EPSON WORLD"
150 PRINT USING "EPSON @ WORLD";"COMPUTER"
160 PRINT USING "#####";1234.56
170 PRINT USING "#####.##";1234.57
180 PRINT USING "+#####.##";123456.789#
190 PRINT USING "#####.##-";-1234.56
200 PRINT USING "**#####.##";123.456
210 PRINT USING "¥¥#####.##";123.456
220 PRINT USING "**¥#####.##";123.146
230 PRINT USING "###,###.##";12345.678#
240 PRINT USING "#####.#^^^^";1234.56
250 PRINT USING "#####._¥";1234.56
260 PRINT USING "#####";123456!
270 END

RUN
E
EPSON
EPSON COMPUTER WORLD
 1235
 1234.57
%+123456.79
 1234.56-
****123.46
   ¥123.46
****¥123.15
 12,345.68
 1234.6E+00
 1235.¥
%123456
OK
```

P

PRINT#
PRINT# USING

【プリント・シャープ】
【プリント・シャープ・ユージング】

PRINT
印刷することから転じて、ファイルに書き込むこと。

PRINT USING
～を用いて、～にしたがってという意味から、書式指定してファイルに書き込むこと。

機　能　シーケンシャルファイルにデータを書き込みます。

書　式

PRINT　#ファイル番号, [式[| , ; |]] ・・・ [USING 書式文字列; [式[| , , |]] ・・・] [| , ; |]

使用例

```
100 OPEN"TEST"FOR OUTPUT AS #1
200 PRINT#1, A$                        ←A$のデータをシーケンシャルファイルに書き込みます。
300 PRINT#1, USING"¥###.##" ; YEN      ←YENのデータを書式指定して書き込みます。
```

解　説

- **ファイル番号**で指定したシーケンシャルファイルに式で与えたデータを書き込みます。PRINT# USING 文は、書式文字列にしたがってデータを書き込みます。
- **式**と**式**の間は、**カンマ(,)**あるいは**セミコロン(;)**で区切ります。**カンマ**で区切ったときはPRINT文と同様に14桁ごとに空白を補いながら書き込みます。**セミコロン**で区切ったときは前のデータに続けて書き込みます。
- PRINT#文、PRINT# USING 文ともに、PRINT文で画面に表示するのと同じ形式でデータを書き込みます。したがって、書き込むデータは適切な区切り記号で区切る必要があります。また、式の後ろに何も付けなければ自動的にCR/LFコードを書き込みます。
  数値データの区切り　空白　カンマ(,)　CRコード(文字コードは13)
  文字データの区切り　カンマ(,)　CRコード

数値データ

- 数値データは空白が区切り記号になるため、式と式の間をセミコロンで区切ります。数値データの後ろには自動的に空白が付加されます。区切り記号にカンマを用いても構いませんが14桁ごとに空白を補いながら書き込むためファイルのデータが冗長になります。

文字データ

• 式と式の間をセミコロンあるいはカンマで区切って並べただけでは、データ中に区切り記号が挿入されません。したがって、プログラムで区切り記号を書き込む必要があります。

読み出せない例

例えば**A$="NAME"，B$="ADDR"**とします。次のステートメント

PRINT#1，A$；B$

は次のようにファイルに書き込みます。

N	A	M	E	A	D	D	R	CR	LF

CR：CR コード
LF：LF コード

次のステートメント

INPUT#，A$，B$

は文字列**"NAMEADDR"**を A$ に読み込み，B$ で読み込むデータがなくなってしまいます。

読み出せる例

2つの別々の文字列として読み出すには PRINT# 文中に次のような区切り文字を挿入します。

PRINT#1，A$；"，"；B$

ファイルには次のように書き込みます。

これによって2つの文字変数にも読み出すことができます。

文字データ自身が、データとしてカンマ、セミコロン、空白をなどを含む場合は、二重引用符でデータを囲って書き込みます。INPUT# 文は次の二重引用符までを一つのデータとして読み取ります。

• PRINT# USING 文の書式文字列の指定方法については PRINT USING 文を参照してください。

• WRITE#文は自動的にデータを二重引用符で囲って出力します。

• 出力するファイルが SCRN：(画面)の場合、漢字などの2バイト文字を正常に表示することはできません。

P

参 照

INPUT#(シーケンシャルファイルからのデータ読み出し)
LINE INPUT#(シーケンシャルファイルからのデータ読み出し)
PRINT USING(書式設定)
WRITE#(シーケンシャルファイルへのデータ書き込み)

PSET
PRESET

【ピー・セット】【ピー・リセット】

Point SET
点を打つという意味から、画面に点を表示すること。

Point RESET
点を打ち直すという意味から、表示した点を消すこと。

機　能　画面の指定した座標に点(ドット)を表示します。

書　式
```
PSET(Wx, Wy) [, 点の色]
PRESET(Wx, Wy) [, 点の色]
```

使用例

```
PEST(100, 100)        ←座標(100, 100)に点を表示します。
PRESET(100, 100)      ←上の位置の点を消去します。
```

解　説
- PSET 文と PRESET 文は、**点の色**を省略した場合の機能が異なるだけで他の内容については共通です。
- 指定したワールド座標(**Wx, Wy**)に**点の色**で指定した色で点(ドット)を表示します。座標の前に STEP を付けて相対座標で指定することもできます。
- 指定した点の座標がビューポート外の場合は何も表示しません。
- **点の色**はパレット番号で指定します。点の色を省略した場合、PSET 文では前景色を、PRESET 文では背景色を指定したことになります。画面モードが白黒モードのときは、点を表示する場合は1、消去する場合は0を指定します。
- 命令実行後、最終参照座標は(**Wx, Wy**)になります。

参　照　COLOR(パレットの設定)

【プット】
PUT
置くと言う意味。ここではデータを書き込むこと。

機　能　ランダムファイルバッファのデータをランダムファイルに書き込みます。

書　式　PUT [#]ファイル番号 [, レコード番号]

使用例　PUT#1, 2　←ファイル番号1のファイルの2番目のレコードへファイルバッファのデータを書き込みます。

解　説
- **ファイル番号**で指定したランダムファイルの指定した**レコード番号**のレコードへファイルバッファの内容を書き込みます。
- **レコード番号**には1から65000までの整数を指定します。**レコード番号**を省略すると、直前のGET#文あるいはPUT#文で指定したレコード番号の次のレコード番号を指定したことになります。ランダムファイルをオープンした直後は1になります。
- ファイルバッファへのデータの設定はLSET, RSET文を使用します。

参　照
FIELD(ランダムファイルのバッファの割り当て)
OPEN(ファイルのオープン)
GET#(ランダムファイルからの読み込み)

P

PUT@

【プット・アットマーク】

PUT
置くと言う意味。ここでは画面にあらかじめ読み取ってある図形を表示すること。

機能　GET@文で読み取ったグラフィックパターンを、画面の指定した位置に表示します。

書式　PUT [@] (Sx, Sy), | 配列変数 [(要素番号)] / KANJI(漢字コード) | [, [機能] [, 前景色, 背景色]]

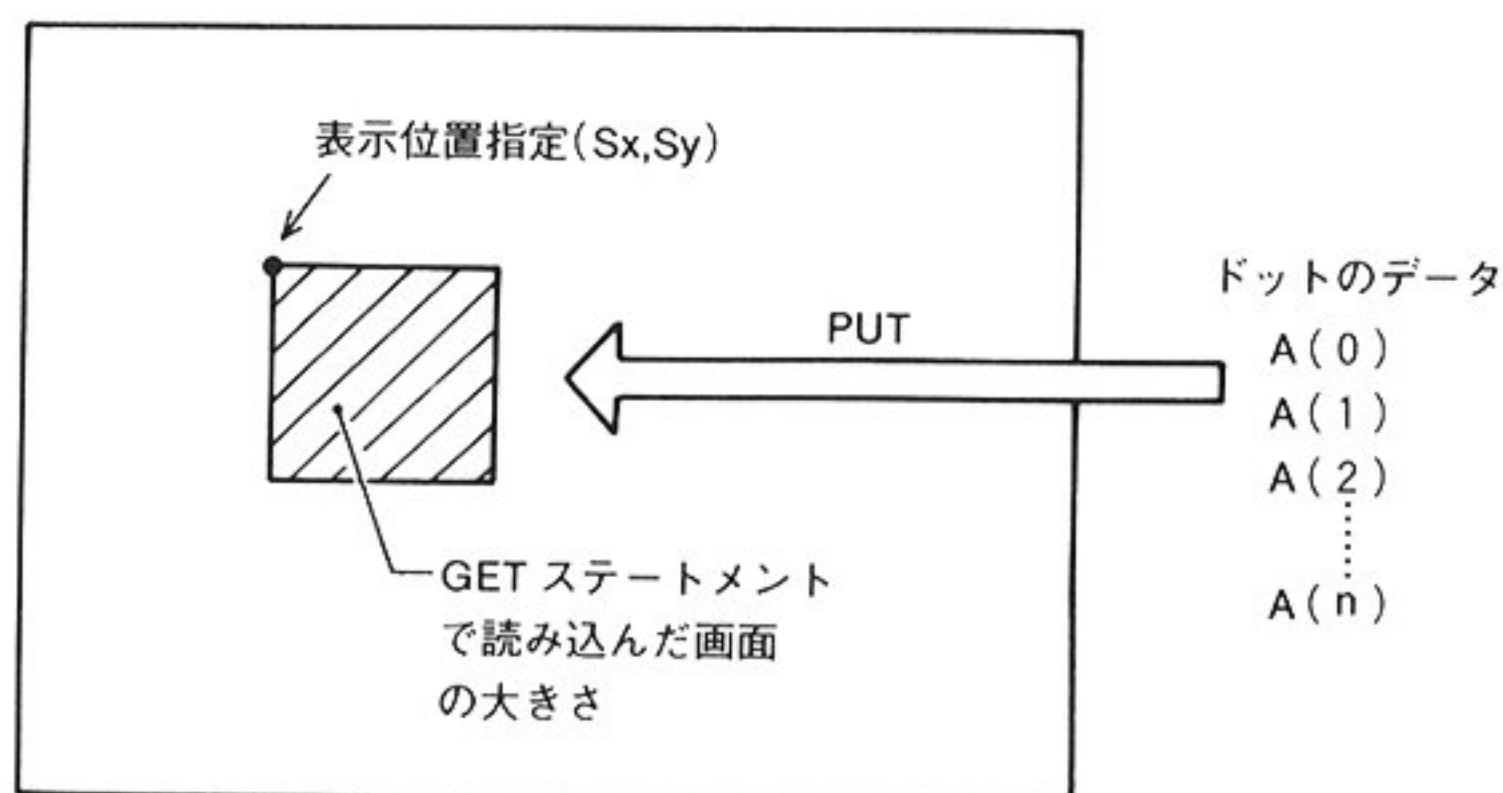
使用例

PUT@(200, 200), G%, PSET　←指定した座標に、配列変数G%のグラフィックパターンを表示します。

PUT@(100, 100), KANJI(&H3441)　←指定した座標に「漢」を表示します。

解説

①配列変数名を指定した場合

- GET@文で**配列変数**に読み込んだグラフィックパターンの左上すみを、スクリーン座標(Sx, Sy)に合わせて表示します。
- **配列変数**は、GET@文でグラフィックパターンを読み込んだ数値型の配列、または相当するデータ形式を持つ数値型の配列名を指定します。
- **要素番号**は、配列内のどの要素から表示するのかを指定します。要素番号を省略した場合は配列の最初(0または1)のデータから表示します。

- **機能**はグラフィックパターンの表示方法を指定します。省略した場合は、XORを指定したことになります。

機能	意　　味
PSET	配列のグラフィックパターンをそのまま表示します。
PRESET	配列のグラフィックパターンを反転して表示します。 白黒モード グラフィックパターンを白黒反転して表示 ８色モード・4096色中８色モード 読み込んだときのパレット番号をビット反転したパレット番号の色で表示 4096色中16色モード 読み込んだときのパレット番号をビット反転したパレット番号の色で表示
AND OR XOR	画面に表示しているパターンと配列のパターンを１ドットずつ対応させ、論理演算を行った結果を表示します。カラーモードの場合は、パレット番号による論理演算になります。したがって表示する色が変わります。

- **前景色**と**背景色**は、白黒モードで読み込んだグラフィックパターンをカラーモードで表示する場合に指定します。この２つは両方同時に指定しなければなりません。

前景色	白黒モードで読み込んだ白(セット)のドットを表示する色をパレット番号で指定します。
背景色	白黒モードで読み込んだ黒(リセット)のドットを表示する色をパレット番号で指定します。

②KANJI(漢字コード)を指定した場合

- 指定したスクリーン座標(**Sx, Sy**)の位置に**漢字コード**で指定した文字を表示します。
- **漢字コード**で指定できる文字はJIS 第１水準，JIS 第２水準およびユーザー定義文字です。
- 漢字パターンには、縦16ドット横16ドット、縦16ドット横８ドット、縦８ドット横８ドットのものがあります。
- 漢字パターンは、配列変数を指定したときと同じように、前景色、背景色を指定してカラー表示することができます。

参　照　GET@(グラフィックパターンの読み込み)

プログラム例

P

```
100 ' PUT@
110 ' --- グラフィック画面に漢字を表示 ---
120 SCREEN 0,0 : CLS 3
130 X=20
140 FOR I=1 TO 10
150   READ A$ : KC=VAL("&H"+A$)
160   PUT(X,30),KANJI(KC),PSET,7,0
170   X=X+50
180 NEXT I
190 DATA 3441,3B7A,2472,493D,3C28
200 DATA 2437,2446,2424,245E,2439

OK
```

漢　字　を　表　示　し　て　い　ま　す

RANDOMIZE 【ランダマイズ】

RANDOMIZE
任意抽出するということから、乱数を発生させること。

機能 RND関数で発生する乱数の乱数シードを設定します。

書式 RANDOMIZE [整数式]

使用例 RANDOMIZE ←乱数シードを設定します。

解説

- **整数式**には−32768から32767までの整数を指定し、乱数シード(乱数発生のための初期値)を設定します。
- **整数式**を省略すると実行を一時停止し、次のような乱数シードを要求するメッセージを表示します。

  Random number seed(−32768 to 32767)?
- RND関数は、乱数シードを変更しないとプログラムを実行するたびに同じ乱数を発生します。RND関数を実行する前にRANDOMIZE文を実行して、異なる乱数列を発生するようにしてください。

参照 RND(乱数の発生)

プログラム例

```
100 ' RANDOMIZE
110 ' ---  時間を利用して乱数のシードを変える  ---
120 '
130 T$=TIME$
140 T=VAL(MID$(T$,7,2))+(VAL(MID$(T$,4,2))+VAL(MID$(T$,1,2))*60)*60
150 T=T-FIX(T/32767)*32767
160 RANDOMIZE T
165 FOR I=1 TO 10
170   PRINT USING "####";RND*100;
190 NEXT

RUN
  71  64  55  91   2  86  73  12  42  74
OK
RUN
  52  98  18   6  48  77  18  30  29  93
OK
RUN
  34  33  81  20  94  67  63  48  15  12
OK
```

READ

【リード】

READ
読むことから、DATA文で定義したデータを読み込むこと。

機能　DATA文で定義しているデータを変数に読み込みます。

書式　READ　変数名 [, 変数名] ・・・

使用例

```
1000 READ A$, B, C$
5000 DATA 11月, 31, 12月
```

←READ文で読み込む定数の型は、DATA文で定義するものと一致させておきます。

解説

- DATA文で定義しているデータを、順次指定した**変数**に読み込んでいきます。したがって、READ文は必ずDATA文と共に使います。
- **変数名**は数値変数でも文字変数でも構いませんが、DATA文で定義しているデータの型と一致しなければなりません。一致しない場合はSyntax error(構文の誤り)エラーになります。ただし、定義しているデータが数字の並びであれば、変数の型はどちらでも構いません。(数値変数であれば数値として、文字変数であれば数字の並んだ文字列として読み込むため)
- READ文は、RESTORE文で指定した行のDATA文のデータから読み込みます。RESTORE文を実行しない場合は、プログラムの最初にあるDATA文のデータから順番に読み込んでいきます。
- 一つのプログラム中で、READ文で読み込むデータの数がDATA文で定義しているデータの数より多い場合は、読み込むデータがなくなった時点でOut of DATA(データがない)エラーになります。

参照

DATA(データの定義)
RESTORE(READ文で読み込むDATA文のある行を設定)

R

REM

【レム】

REMark
注釈や注意書きのこと。

機能　プログラムに注釈や、説明を入れてプログラムをわかりやすくします。

書式

```
REM　文字列
'　　文字列
```

使用例

```
100　REM メインメニュー
100　'サブルーチン
```

←プログラムの途中などで、実行する機能やサブルーチンであることを明確にします。

解説

- REM 文はプログラムの実行になんの影響も与えない非実行文です。REM 文を使うとプログラムをわかりやすく、見やすくすることができます。
- REM 文の後ろに続く**文字列**はすべて非実行文になります。したがって、同一行で REM 文より後ろに実行文を続けても、実行することはできません。
- 文字列には漢字やひらがななどの2バイト文字を使うこともできます。
- REM 文の代わりにアポストロフィ(')で代用することができます。

RENUM

【リナンバー】

RENUMber
再び番号を付けることから、行番号を付け直すこと。

機 能　プログラムの行番号を付け直します。

書 式　RENUM　[新行番号] [, [旧行番号] [, 増分]]

使用例

```
RENUM              ←新しい行番号は10, 20, 30・・・になります。
RENUM 100, 10, 20  ←新しい行番号は100, 120, 140, ・・・になります。
```

解 説

- **新行番号**は、新しく付け直す行番号の最初の行番号を指定します。省略すると10を指定したことになります。
- **旧行番号**は行番号の付け直しを開始する現在のプログラムの行番号を指定します。省略するとプログラムの最初の行番号を指定したことになります。
- **増分**は、新しく付け直す行番号の増分値です。省略すると10を指定したことになります。
- RENUM コマンドはGOTO, GOSUB, THEN, ON～GOTO, ON～GOSUB, RESTORE, RESUME, ERL などで指定している行番号も、自動的に新しい行番号に対応して変更します。これらの命令で指定している行番号が、プログラム中に存在しなかった場合は **Undefined line xxxxx in yyyyy**(yyyyy 行で参照している xxxxx 行が存在しない)を表示し、誤った行番号(xxxxx)がそのまま残ります。
- プログラム行の順番を変えるような指定はできません。
  例：プログラムが10, 20, 30・・・と行を持つとき、RENUM 15, 30 を実行して行を入れ替えることはできません。
- 65529より大きな行番号が必要になる行番号の付け直しは Illegal function call(違法関数呼び出し)エラーとなり、付け直し作業を行いません。
- 旧番号にピリオド(.)を指定すると、直前に参照した行番号を指定したことになります。

R

RESTORE

【リストア】

RESTORE
もとに戻す、もとどおりにするという意味から、データの読み込みを再び行うこと。

機　能　READ 文で読み込む、DATA 文のある行番号を指定します。

書　式　RESTORE　[行番号]

使用例

```
100  RESTORE  300
110  READ  A, B, C
200  DATA  1, 2, 3
300  DATA  7, 8, 9
```

←A, B, C に読み込むデータを300行に指定します。

解　説

- **行番号**は次に実行する READ 文で読み込む、DATA 文のある行番号を指定します。
- **行番号**を省略すると、プログラム中の最も最初にある DATA 文のある行を指定したことになります。

参　照　DATA（READ文で読み込むデータの定義）
READ（データの読み込み）

RESUME

【リジューム】

RESUME
回復する、再開するという意味。プログラムの実行を再開すること。

機能 エラー処理ルーチンから、もとのプログラムに戻ります。

書式

```
RESUME | [0]   |
       | NEXT  |
       | 行番号 |
```

使用例

RESUME 0	←エラーの発生した文に戻ります。
RESUME NEXT	←エラーの発生した次の文に戻ります。
RESUME 50	←50行に制御を移します。

解説

- ON ERROR GOTO 文で定義したエラー処理ルーチンから、もとのプログラム(エラーの発生する前に実行していたプログラム)に戻ります。
- 戻る場所は次のとおりです。

書式	戻る場所
RESUME [0]	エラーの発生した行に戻ります。0は省略することができます。
RESUME NEXT	エラーの発生した次の文に戻ります。
RESUME 行番号	行番号で指定した行に戻ります。

参照 ON ERROR GOTO(エラー処理ルーチンの定義)

R

RETURN

【リターン】

RETURN
帰る、戻ることから、もとのプログラムに戻ること。

機 能　サブルーチンからもとのプログラムへ戻ります。

書 式　RETURN　[行番号]

使用例

RETURN	←サブルーチンを呼び出した行の次の文に戻ります。
RETURN 100	←100行に戻ります。

解 説

- GOUSUB 文から実行するサブルーチンや ON KEY GOSUB 文などの割り込み処理ルーチンから呼び出したプログラム(もとのプログラム)へ戻るときに使用します。
- 通常は**行番号**を指定せずに使用します。この場合、次の場所に戻ります。

  GOSUB/ON～GOSUB

  その文のある次の文に戻ります。

  ON KEY GOSUB/ON COM GOSUB/・・・・・などの割り込み処理ルーチン

  割り込み要因が発生した直後の文に戻ります。割り込みの発生は、プログラムの文を一つ実行するたびにチェックしており、そのとき割り込みの要因があると割り込み処理ルーチンを実行します。したがって RETRUN 文では次の命令から実行することになります。
- **行番号**を指定すると、サブルーチンを呼び出した行に関係なく、指定した行に戻ることができます。行番号を指定する場合は、直接 FOR 文や WHILE 文のループ内を実行したり、GOSUB 文を実行せずにサブルチーン内を実行することのないよう注意してください。
- サブルーチンの実行なしに RETURN 文を実行すると RETURN without GOSUB (RETURN 文の数が多い)エラーが発生します。

参 照　GOSUB(サブルーチンの呼び出し)
各種割り込み処理ルーチンの呼び出し

RIGHT$

【ライト・ダラー】

RIGHT
right は右側のこと。

機　能　文字列の右側から指定した文字数分の文字列を返します。

書　式　RIGHT$(文字列, 文字数)

使用例　A$=RIGHT$(B$, 5)　←文字列 B$ の右側から 5 文字分を A$ に代入します。

解　説
- 指定した**文字列**の右側から、**文字数**で指定するバイト数分の文字列を返します。
- **文字数**は、0 から255までの範囲で、取り出す文字列のバイト数を指定します。**文字数**が文字列全体のバイト数より大きいときは、その文字列のすべてを取り出します。また、**文字数**が 0 であれば空文字列("")を返します。
- もとの**文字列**の内容は変化しません。

参　照　LEFT$(文字列の左側から文字列を取り出す)
MID$(文字列の指定位置から文字列を取り出す)

プログラム例

```
100 ' RIGHT$
110 ' ---  右からの文字列を読み取る  ---
120 A$="ABCDEFGHIJ"
130 FOR I=1 TO 10
140   B$=RIGHT$(A$,I)
150   PRINT B$
160 NEXT I
170 END

RUN
J
IJ
HIJ
GHIJ
FGHIJ
EFGHIJ
DEFGHIJ
CDEFGHIJ
BCDEFGHIJ
ABCDEFGHIJ
OK
```

R

RND

【ランド】

RaNDom
でたらめの意味。乱数(random number)のこと。

機 能　0以上1未満の間の乱数を発生します。(0≦RND<1)

書 式　RND [(機能)]

使用例　A=RND(1)　　←次の乱数を発生します。

解 説
- **機能**にしたがった乱数を発生します。**機能**を省略した場合は、正の数を指定したことになります。

機能	意　味
正の数	同じ乱数系列の中の次の乱数を発生します。
0	同じ乱数系列の中の一つ前の乱数を発生します。
負の数	同じ乱数系列の最初の乱数を発生します。(初期化)

- RND関数はRUN, CLEAR文を実行するたびに同じ乱数を発生します(同じ乱数系列)。実行のたびに違った乱数を発生させるには、RANDOMIZE文を実行し、乱数系列を変更するようにプログラムを作成してください。

参 照　RANDOMIZE(乱数シードの設定)

ROLL

【ロール】

ROLL
巻くという意味。巻き紙を巻くように　画面の表示を変えること。

機　能　グラフィック画面を上下あるいは左右にスクロールします。

書　式　ROLL　[上下方向ドット数] [, [左右方向ドット数] [, [|N|Y|]]]

使用例　ROLL 16　←画面を16ドット上方向にスクロールします。

解　説

- グラフィック画面を、指定したドット数分だけ上下左右方向にスクロールします。
- **上下方向ドット数**は上下方向にスクロールするドット数で、画面のモードにより指定できる範囲は異なります。
  標準モード(640×200ドット)：－199から199
  高解像モード(640×400ドット)：－399から399
  正の数を指定した場合は上方向に、負の数を指定した場合は下方向にドット数(絶対値)だけスクロールします。省略した場合はスクロールを行いません。
- **左右方向ドット数**は左右方向にスクロールするドット数で、画面の横ドット数を指定します。指定できる範囲は－639から639です。正の数を指定した場合は左方向に、負の数を指定した場合は右方向に、ドット数(絶対値)以下で最も大きい８の倍数のドット数だけスクロールします。省略した場合はスクロールを行いません。
- **N**または**Y**はスクロールによって消えた画面の色を指定するもので次の意味になります。

N	スクロールによって消えた領域をパレット番号０の色で塗りつぶします。
Y	スクロールによって消えた領域を背景色で塗りつぶします。

省略した場合はＮを指定したことになります。

R

RUN

【ラン】

RUN
働く、走るという意味からプログラムを実行すること。

機　能

メモリ中のBASICプログラムを実行します。また、ディスクからプログラムをロードして実行することもできます。

書　式

RUN　[行番号]
RUN　ファイル指定子［,R］

使用例

RUN 100　←メモリ中のプログラムの100行から実行します。

RUN "DEMO"　←ドライブ1からプログラムDEMOをロードして実行します。

解　説

・指定したプログラムを実行します。

RUN［行番号］

・**行番号**を指定すると、メモリ中のプログラムのその行から実行を始めます。**行番号**を省略すると、プログラムの先頭から実行を開始します。

・プログラム実行後はBASICのコマンドレベルに戻ります。

RUN　ファイル指定子［,R］

・**ファイル指定子**で指定したプログラムを読み込んで実行します。

・Rを指定をするとすでにオープンしているファイルをクローズせずにプログラムを実行します。これは**LOADファイル指定子,R**と同機能になります。R指定がない場合は、すべてのファイルをクローズしてプログラムの実行を開始します。

参　照

CHAIN（プログラムの連鎖実行）
LOAD（プログラムの読み込み）

SAVE

【セイブ】

SAVE
蓄える、取っておくの意味からプログラムを保存すること。

機能　メモリ中のBASICプログラムをディスクに保存(セーブ)します。

書式　SAVE　ファイル指定子 [, |A|]
|P|

使用例

```
SAVE "TEST. BAS"
```

←TEST. BASというファイル名でセーブします。

解説

- **ファイル指定子**で指定したファイル名でメモリ中のプログラムを保存します。これをプログラムをセーブするといいます。
- **ファイル指定子**に続けてAまたはPの指定を行うことができます。

省略	プログラムを通常の形式(バイナリ形式)でセーブします。この方法が一般的です。
A指定	アスキー形式でセーブします。アスキー形式は画面に表示するのと同じ形式でセーブします。バイナリ形式のセーブより大きいディスク上の領域を必要とします。
P指定	プロテクト形式でセーブします。このセーブはプログラム保護のための機能で、セーブしたプログラムの内容を変更することができなくなります。プログラムをロードし、LISTやEDITを実行しようとするとIllegal function call(違法関数呼び出し)エラーになり、内容の確認や修正を行うことはできません。

注) このようにプロテクト形式でセーブしたプログラムは再度修正することができなくなります。また、この指定を解除する方法もありません。したがって不用意に使用しないよう注意してください。

参照　LOAD(プログラムの読み込み)

S

SCREEN

【スクリーン】

SCREEN

screen とは画面のこと。ここでは画面モードを設定すること。

機　能　グラフィック画面の表示モードを設定します。

グラフィック画面の精度

グラフィック画面の表示の有無

グラフィック画面の描画と表示

書　式　SCREEN　[画面モード] [, [画面スイッチ] [, [描画ページ] [, [表示ページ]]]]

使用例　SCREEN 3　　←グラフィック画面を高解像カラーモードに設定します。

解　説

• **画面モード**には0から3までの整数を指定し、画面の種類を設定します。

指定値	画面モード	画面の精度	ページ数
0	標準カラーモード	640×200 ドット	4
1	標準白黒モード	640×200 ドット	16(12)
2	高解像白黒モード	640×400 ドット	8(6)
3	高解像カラーモード	640×400 ドット	2

ページ数は同じ大きさの画面の数を示します。(　)内の数値は基本グラフィックモードでの画面数です。ページはそれぞれ1から始まる数値がつけられています。

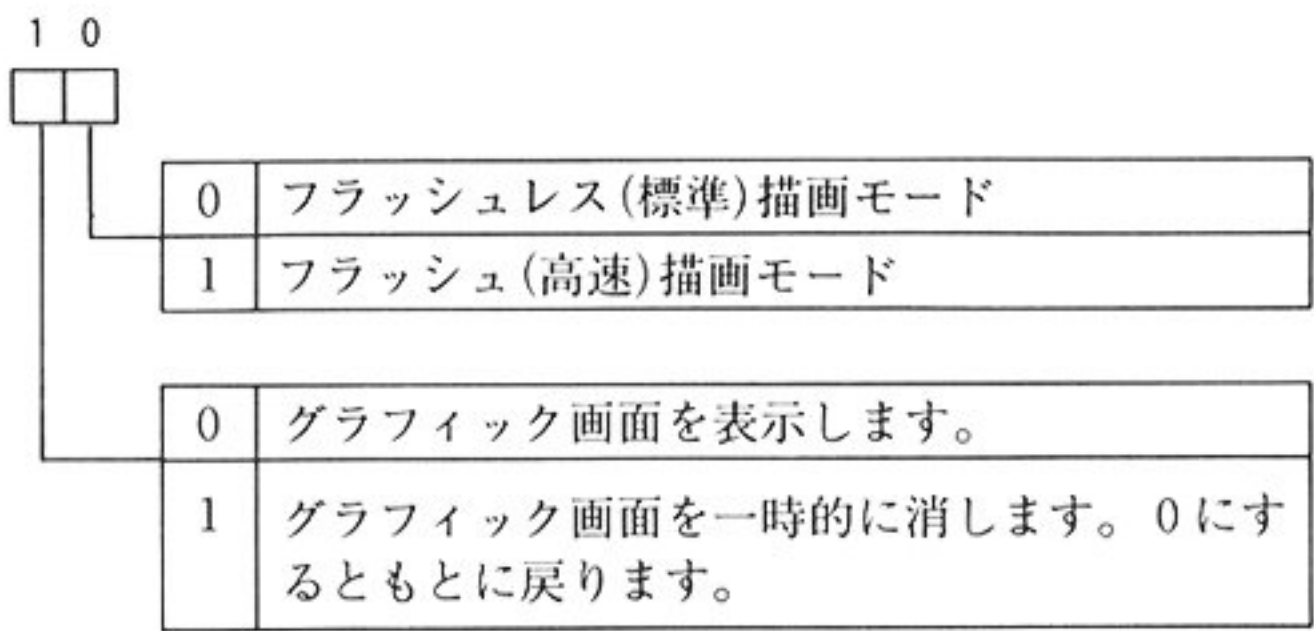
• **画面スイッチ**は2ビットのスイッチから構成され0から3までの値を指定します。それぞれのスイッチの意味は次のとおりです。

1 0

ビット0:

0	フラッシュレス(標準)描画モード
1	フラッシュ(高速)描画モード

ビット1:

0	グラフィック画面を表示します。
1	グラフィック画面を一時的に消します。0にするともとに戻ります。

• **描画ページ**と**表示ページ**は、いくつか持っているページに対して、グラフィックを描くページと、表示するページを指定します。したがって、描画ページと表示ページが一致しないと、内部的にグラフィックを描き、画面に表示しないことができます。描画ページと表示ページで指定できる範囲は各画面モードにより異なります。

(1)　**8色中・8色モードまたは4096色中・8色モードの場合**

・**描画ページ**で指定できる値は次のとおりです。

画面モード ＼ 指定値	標準カラーモード	標準白黒モード	高解像カラーモード	高解像白黒モード
0	1	1	1	1
1	2	2	2	2
2	3	3	—	3
3	4	4	—	4
4	—	5	—	5
5	—	6	—	6
6	—	7	—	—
7	—	8	—	—
8	—	9	—	—
9	—	10	—	—
10	—	11	—	—
11	—	12	—	—

・**表示ページ**で指定するページの関係は次のとおりです。

画面モード ＼ 指定値	標準カラーモード	標準白黒モード			高解像カラーモード	高解像白黒モード		
0	—	—			—	—		
1	**1**	**1**	2	3	**1**	**1**	2	3
2	**2**	1	**2**	3	×	1	**2**	3
3	×	**1**	**2**	3	×	**1**	**2**	3
4	×	1	2	**3**	×	1	2	**3**
5	×	**1**	2	**3**	×	**1**	2	**3**
6	×	1	**2**	**3**	×	1	**2**	**3**
7	×	**1**	**2**	**3**	×	**1**	**2**	**3**
8	—	—			—	—		
9	×	**4**	5	6	×	×		
10	×	4	**5**	6	×	×		
11	×	**4**	**5**	6	×	×		
12	×	4	5	**6**	×	×		
13	×	**4**	5	**6**	×	×		
14	×	4	**5**	**6**	×	×		
15	×	**4**	**5**	**6**	×	×		
16	—	—			—	—		
17	**3**	**7**	8	9	**2**	**4**	5	6
18	**4**	7	**8**	9	×	4	**5**	6
19	×	**7**	**8**	9	×	**4**	**5**	6
20	×	7	8	**9**	×	4	5	**6**

S

画面モード / 指定値	標準カラーモード	標準白黒モード			高解像カラーモード	高解像白黒モード		
21	×	**7**	8	**9**	×	**4**	5	**6**
22	×	7	**8**	**9**	×	4	**5**	**6**
23	×	**7**	**8**	**9**	×	**4**	**5**	**6**
24	—	—			—	—		
25	×	**10**	11	12	×	×		
26	×	10	**11**	12	×	×		
27	×	**10**	**11**	12	×	×		
26	×	10	11	**12**	×	×		
29	×	**10**	11	**12**	×	×		
26	×	10	**11**	**12**	×	×		
31	×	**10**	**11**	**12**	×	×		

×：指定できない
−：全画面表示しない
□：表示画面

(2) **4096色中・16色モードの場合**

• **描画ページ**で指定できる値は次のとおりです。

画面モード / 指定値	標準カラーモード	標準白黒モード	高解像カラーモード	高解像白黒モード
0	1	1	1	1
1	2	2	2	2
2	3	3	—	3
3	4	4	—	4
4	—	5	—	5
5	—	6	—	6
6	—	7	—	7
7	—	8	—	8
8	—	9	—	—
9	—	10	—	—
10	—	11	—	—
11	—	12	—	—
12	—	13	—	—
13	—	14	—	—
14	—	15	—	—
15	—	16	—	—

• 表示ページで指定する値と実際に表示するページの関係は次のとおりです。

画面モード＼指定値	標準カラーモード	標準白黒モード				高解像カラーモード	高解像白黒モード			
0	−	−				−	−			
1	**1**	**1**	2	3	4	**1**	**1**	2	3	4
2	**2**	1	**2**	3	4	×	1	**2**	3	4
3	×	**1**	**2**	3	4	×	**1**	**2**	3	4
4	×	1	2	**3**	4	×	1	2	**3**	4
5	×	**1**	2	**3**	4	×	**1**	2	**3**	4
6	×	1	**2**	**3**	4	×	1	**2**	**3**	4
7	×	**1**	**2**	**3**	4	×	**1**	**2**	**3**	4
8	×	1	2	3	**4**	×	1	2	3	**4**
9	×	**1**	2	3	**4**	×	**1**	2	3	**4**
10	×	1	**2**	3	**4**	×	1	**2**	3	**4**
11	×	**1**	**2**	3	**4**	×	**1**	**2**	3	**4**
12	×	1	2	**3**	**4**	×	1	2	**3**	**4**
13	×	**1**	2	**3**	**4**	×	**1**	2	**3**	**4**
14	×	1	**2**	**3**	**4**	×	1	**2**	**3**	**4**
15	×	**1**	**2**	**3**	**4**	×	**1**	**2**	**3**	**4**
16	−	−				−	−			
17	×	**5**	6	7	8	×	×			
18	×	5	**6**	7	8	×	×			
19	×	**5**	**6**	7	8	×	×			
20	×	5	6	**7**	8	×	×			
21	×	**5**	6	**7**	8	×	×			
22	×	5	**6**	**7**	8	×	×			
23	×	**5**	**6**	**7**	8	×	×			
24	×	5	6	7	**8**	×	×			
25	×	**5**	6	7	**8**	×	×			
26	×	5	**6**	7	**8**	×	×			
27	×	**5**	**6**	7	**8**	×	×			
28	×	5	6	**7**	**8**	×	×			
29	×	**5**	6	**7**	**8**	×	×			
30	×	5	**6**	**7**	**8**	×	×			
31	×	**5**	**6**	**7**	**8**	×	×			
32	−	−				−	−			
33	**3**	**9**	10	11	12	**2**	**5**	6	7	8
34	**4**	9	**10**	11	12	×	5	**6**	7	8
35	×	**9**	**10**	11	12	×	**5**	**6**	7	8
36	×	9	10	**11**	12	×	5	6	**7**	8
37	×	**9**	10	**11**	12	×	**5**	6	**7**	8
38	×	9	**10**	**11**	12	×	5	**6**	**7**	8
39	×	**9**	**10**	**11**	12	×	**5**	**6**	**7**	8
40	×	9	10	11	**12**	×	5	6	7	**8**

S

指定値＼画面モード	標準カラーモード	標準白黒モード				高解像カラーモード	高解像白黒モード			
41	×	**9**	10	11	**12**	×	**5**	6	7	**8**
42	×	9	**10**	11	**12**	×	5	**6**	7	**8**
43	×	**9**	**10**	11	**12**	×	**5**	**6**	7	**8**
44	×	9	10	**11**	**12**	×	5	6	**7**	**8**
45	×	**9**	10	**11**	**12**	×	**5**	6	**7**	**8**
46	×	9	**10**	**11**	**12**	×	5	**6**	**7**	**8**
47	×	**9**	**10**	**11**	**12**	×	**5**	**6**	**7**	**8**
48	—	—				—	—			
49	×	**13**	14	15	16	×	×			
50	×	13	**14**	15	16	×	×			
51	×	**13**	**14**	15	16	×	×			
52	×	13	14	**15**	16	×	×			
53	×	**13**	14	**15**	16	×	×			
54	×	13	**14**	**15**	16	×	×			
55	×	**13**	**14**	**15**	16	×	×			
56	×	13	14	15	**16**	×	×			
57	×	**13**	14	15	**16**	×	×			
58	×	13	**14**	15	**16**	×	×			
59	×	**13**	**14**	15	**16**	×	×			
60	×	13	14	**15**	**16**	×	×			
61	×	**13**	14	**15**	**16**	×	×			
62	×	13	**14**	**15**	**16**	×	×			
63	×	**13**	**14**	**15**	**16**	×	×			

×：指定できない
－：全画面表示しない
□：表示画面

- SCREEN 文を実行するとウインドウ、ビューポート、最終参照座標の設定は初期状態に戻ります。

参　照　COLOR（カラーモードの設定）

SEARCH

【サーチ】
SEARCH
探すという意味。

機能　指定した値の、配列変数中での位置を与えます。

書式　SEARCH(配列変数名, データ [, [要素番号] [, [増分]]])

使用例　A＝SEARCH(B%, 100, 0, 2)　←配列B%の中を最初から2ステップおきに100を探し、その添字をAに代入します。

解説
- 指定した**配列変数**の中で、指定した**データ**を記憶している最初の要素番号(添字)を返します。指定した**データ**が見つからない場合は－1を返します。
- 指定できる**配列変数**は、整数型の1次元配列です。
- **要素番号**は、指定したデータを探し始める添字の番号です。**要素番号**を省略するとOPTION BASE文によって設定している添字の下限(0または1)の要素から探し始めます。
- **増分**は、調べる要素番号を指定するもので、現在の要素番号に増分を加えたものが、次に調べる要素になります。省略した場合は1を指定したことになります。

S

SET

【セット】

SET
～の状態にする意味から、ここではファイルの状態を設定すること。

機　能　ファイルの属性を設定します。

書　式

```
SET  ドライブ番号      , "[P][R]"
     ファイル指定子
     #ファイル番号
```

使用例

SET 1, "R"	←ドライブ1に入っているディスクにリードアフタライトの属性を設定します。
SET "NECESS. ITY", "P"	←ファイル NECESS. ITY を書き込み禁止にします。
SET #1, "P"	←ファイル# 1を書き込み禁止にします。

解　説

• 指定した**ドライブ**中の**ディスク**や**ファイル**にPやRの属性を設定します。それぞれの属性の意味は次のとおりです。

P	指定したディスク、ファイルに対して書き込みを禁止します。したがってPRINT# 文などでデータの書き込みをしようとすると File write protected エラーになります。また KILL 文によるファイルの削除もできなくなります。
R	ファイルに書き込みを行った直後に読み出しを行い、その書き込みが正しいかどうかの確認を行います。

• SET 文は、最初にすべての属性を解除し、その後、R、Pで指定した属性を設定します。P, R以外の文字を使用した場合は、何も行いません。

• **ドライブ番号**を指定した場合は、そのドライブのディスクに対して属性が設定されます。P指定をした場合は、このディスクに対してファイルを新しく作成したり、削除することはできません。

• **ファイル指定子**を指定した場合は、そのファイルに対してのみ属性を設定します。ほかのファイルには影響しません。

• **ファイル番号**を指定した場合は、そのファイルをオープンしている間だけ、その属性が働きます。

参　照　ATTR$(ドライブ、ファイルの属性を調べる)

SGN

【サイン】
SiGN
符号の意味。

（機　能）　数値が正の数のときは1、0のときは0、負の数のときは－1を与えます。

（書　式）　SGN(数式)

（使用例）　PRINT SGN(－256)　　←結果は－1です。

（解　説）
- (　)内に指定した**数式**の(計算後の値の)符号を調べます。
- SGN関数の値は、次のとおりです。

数式＞0のとき	1
数式＝0のとき	0
数式＜0のとき	－1

- ABS(X)＊SGN(X)は必ずXと等しくなります。

（参　照）　ABS(絶対値の計算)

S

SIN

【サイン】

SINe
三角関数の sin(正弦)の意味。

機　能　正弦(サイン)を計算します。

書　式　SIN(数式)

使用例
```
PI#=3.14159265358979323846
A=SIN(30*PI#/180)                ←30°の正弦を計算します。
```

解　説
- ( )内の**数式**の正弦を計算します。数値の単位はラジアンです。角度を度で与える場合は、ラジアンに変換するためにπ/180を掛けます。πには次のような値を使用すると最も精度の高い計算を行うことができます。
  倍精度実数(PI# をπの値を持つ変数とします)
  PI#=3.14159265358979323846
- **数式**が倍精度実数のときは、結果も倍精度で計算した値になります。他の場合は、単精度で計算した値になります。

参　照
ATN(逆正接)
COS(余弦)
TAN(正接)

SPACE$

【スペース・ダラー】

SPACE
空白のこと。空白の文字列を作ること。

機　能　空白(文字コード32)を繰り返す文字列を作ります。

書　式　SPACE$(空白の数)

使用例　A$=SPACE$(50)　←変数 A$ は空白が50個の文字列です。

解　説

- **空白の数**だけ空白(" ")を続けた文字列を作ります。指定できる数値は 0 から255までの整数です。

参　照　SPC(空白を出力)
STRING$(指定した文字を繰り返した文字列を作る)

S

SPC

【スペース】

SPaCe
空白のこと。余白が必要なときに用いる。

機能　指定の数だけ空白を画面やプリンタに出力します。

書式　SPC(空白の数)

使用例　PRINT "LEFT";SPC(10);"RIGHT"　　←2つの単語の間に10個の空白を挿入します。

解説

- **空白の数**の空白(" ")を画面やプリンタに出力します。
- SPC関数は単独では使用しません。必ず、PRINT，PRINT USING，LPRINT，LPRINT USING，PRINT＃，PRINT ＃ USING文と共に使用します。
- **空白の数**には－32768から32767までの数値を指定することができます。ただし、値が画面やプリンタの1行の文字数を超えている場合は、数値を桁数で割った余りを空白の数にします。また、負の数を指定した場合は、0を指定したことになります。
- PRINT文の終わりにSPC関数を指定すると、空白の後ろにセミコロン(;)を付けた場合と同じく、復帰/改行を行いません。

参照　SPACE$(空白の文字列を作る)

プログラム例

```
100 ' SPC
110 '----  スペースを画面に表示する  ----
120 WIDTH 80,25
130 PRINT "    +····+····+"
140 PRINT "LEFT"SPC(10)"RIGHT"
150 PRINT "LEFT"SPC(10)
160 PRINT "RIGHT"
170 PRINT "LEFT"SPC(90)"RIGHT"
180 END

RUN
    +····+····+
LEFT          RIGHT
LEFT          RIGHT
LEFT          RIGHT
OK
```

SQR

【スクェア・ルート】

SQuare Root
square root で平方根という意味。

機　能　平方根(√￣)の値を計算します。

書　式　SQR(数式)

使用例　A=SQR(3)　　← $\sqrt{3}$を計算します。

解　説
- ( )内の**数式**の平方根を計算します。
- **数式**が倍精度実数のときは、結果も倍精度で計算した値になります。他の場合は、単精度で計算した値になります。

STOP

【ストップ】

STOP
停止の意味。プログラムの実行を一時停止すること。

機能 プログラムの実行を一時停止して、コマンドレベルに戻ります。

書式 STOP

使用例 1200　STOP　　←1200行でプログラムの実行が停止します。

解説

- プログラムの実行を一時停止します。プログラム中のどこにでも、いくつでも置くことができます。
- STOP文を実行すると、次のメッセージを表示し、コマンドレベルに戻ります。
  Break in XXXX（XXXXはSTOP文のある行の行番号）
- END文とは異なりオープンしているファイルをクローズしないでプログラムを停止します。また、それまで実行した変数の値など、すべてそのままの状態に保ちます。
- STOP文でプログラムを一時停止し、ダイレクトモードで変数の値を調べ、CONTコマンドで実行を再開すれば、プログラムのデバッグに役立ちます。

参照 CONT（プログラムの実行再開）
END（プログラムの終了）

STOP ON
STOP OFF
STOP STOP

【ストップ・オン】
【ストップ・オフ】
【ストップ・ストップ】
STOP key ON/OFF/STOP

STOPはここではSTOPキーのこと。STOPキーに関する制御を行うこと。

機能　[STOP]キーを押すと発生する割り込み処理ルーチンの実行を許可、禁止、停止します。

書式

```
STOP |ON  |
     |OFF |
     |STOP|
```

使用例

STOP ON	←割り込み処理を実行します。
STOP OFF	←割り込み処理を禁止します。
STOP STOP	←割り込み処理を停止します。

解説

- [STOP]キー(および[CTRL]+[C]キー)による割り込み処理の制御を行います。
- **STOP ON**は[STOP]キーによる割り込み処理を許可します。この命令を実行後は、[STOP]キーを押すごとに割り込みが発生し、ON STOP GOSUB文で指定した割り込み処理ルーチンを実行します。
- **STOP OFF**は[STOP]キーによる割り込み処理を禁止します。この命令を実行後は、[STOP]キーを押しても割り込みは発生せず、[STOP]キー本来の動作をします。プログラム終了時には必ず、STOP OFFを実行してください。
- **STOP STOP**は[STOP]キーによる割り込み処理を停止します。この命令を実行後は、[STOP]キーが押されたことだけを記憶し、割り込み処理は実行しません。その後、STOP ONを実行すると、先ほど[STOP]キーが押されたことによって割り込み処理ルーチンを実行します。

参照　ON STOP GOSUB(STOPキーによる割り込み処理ルーチンの定義)

S

STR$

【ストリング・ダラー】

STRing
一列の意味。ここでは文字列のこと。

機　能　数値を画面に表示する形式の文字列にします。

書　式　STR$(数式)

使用例　A$=STR$(123)　　←123をそのまま文字列" 123"としてA$に代入します。

解　説
- **数値**を、PRINT文で画面に表示する形式の文字列に変換します。
- **数値**が正の数のときは、先頭に空白1文字(プラス記号のためにあけている)を付加します。負の数のときはマイナス記号(−)を含みます。

参　照　VAL(文字列を数値に変換)

プログラム例

```
100 ' STR$
110 ' ---  数値を文字列として扱う  ---
120 INPUT M,N
130 A$=STR$(M) : B$=STR$(N)
140 PRINT M+N
150 PRINT A$+B$
160 GOTO 120

RUN
? 12,34
 46
 12 34
?
```

STRING$

【ストリング・ダラー】

STRING
一列の意味。ここでは文字列のこと。

（機　能）　同じ文字を、いくつか続けた文字列を作ります。

（書　式）　STRING$(文字数, |文字列
文字コード|)

（使用例）

```
A$=STRING$(50, 65)      ←Aを50文字並べた文字列を作ります。
A$=STRING$(50, "A")     ←同じくAを50文字並べた文字列を作ります。
```

（解　説）

- **文字列**の先頭1文字(1バイト文字だけ)、あるいは指定した**文字コード**を持つ文字を**文字数**分続けた文字列を作ります。
- **文字数**は、続ける文字の数で0から255までの数値を指定します。
- **文字列**を指定した場合は、その先頭の1文字を指定したことになります。また**文字コード**には0から255までの数値を指定します。
- 漢字などの2バイト文字を指定することはできません。

（参　照）　SPACE$(空白の文字列を作る)

S

SWAP

【スワップ】

SWAP
交換の意味。変数の値を入れ替えること。

機　能　2つの変数の値を交換(入れ替え)します。

書　式　SWAP　変数1，変数2

使用例　IF A(I)>A(J) THEN SWAP A(I),A(J)　←A(I)>A(J)のとき値を交換します。

解　説
- **変数1**と**変数2**で指定した変数の値を交換します。
- どんな型の変数(整数、単精度実数、倍精度実数、文字列)でも指定できますが、2つの変数の型は一致していなければなりません。2つの型が一致しない場合はType mismatch(型の不一致)エラーになります。

プログラム例

```
100 ' SWAP
110 ' ---  バブルソート  ---
120 C=100 : DIM A(C)
130 ' -- 乱数で数字を 100個発生 --
140 FOR I=1 TO C
150   A(I)=RND*100
160 NEXT
170 ' -- 昇順に並び換え --
180 FOR I=1 TO C-1
190   FOR J=I+1 TO C
200     IF A(I)>A(J) THEN SWAP A(I),A(J)
210   NEXT J
220 NEXT I
230 FOR I=1 TO C
240   PRINT USING "####";A(I);
250 NEXT
260 END

RUN
   1   6   6   7   7   7  10  11  11  12  13  14  16  17  18  19
  24  26  28  29  32  32  33  33  34  35  36  38  38  41  42  43
  46  47  48  48  49  51  52  53  53  53  54  56  56  56  57  57
  62  65  66  66  67  67  68  69  70  70  71  72  72  73  73  74
  80  80  81  82  83  85  85  87  89  91  93  93  93  94  95  95
OK
```

TAB

【タブ】

TABulate
表にすること。ここでは表作成などに用いる桁方向の空白のこと。

機能　文字を表示する桁位置を指定します。

書式　TAB(桁位置)

使用例　PRINT TAB(10);N$　　←11桁目から N$ を表示します。

解説
- **桁位置**で指定する位置からデータを表示します。
- TAB関数は、PRINT，LPRINT 文などの出力命令の中で使用します。
- **桁位置**には－32768から32767の範囲の整数を指定しますが、負の数はすべて0とみなします。また、0は左端を意味します。**桁位置**が画面やプリンタの1行の桁数を超える場合は、**桁位置**を一行の桁数で割った余りが実際の桁位置になります。
- 漢字などの2バイト文字に対してTAB関数を使用することはできません。
- 現在のカーソルの桁位置が指定した桁位置より右にある場合、次の行の指定桁位置に文字を表示します。

参照　SPC(空白の出力)

プログラム例

```
100 ' TAB
110 ' ---  文字を表示する桁位置の指定  ---
120 PRINT "0123456789012345678901234567890123456789"
130 FOR I=1 TO 5
140   READ A,B$
150   PRINT TAB(A);B$
160 NEXT
170 DATA 0,*,5,*,15,*,20,*,38,*
180 END

RUN
0123456789012345678901234567890123456789
*
     *
               *
                    *
                                      *
OK
```

T

TAN

【タンジェント】

TANgent
三角関数の tan(正接)の意味。

機能　正接(タンジェント)を計算します。

書式　TAN(数式)

使用例
```
PI#=3.14159265358979323846        ←30°の正接を計算します。
A=TAN(30*PI#/180)
```

解説
- ( )内の**数式**の正接値を計算します。**数値**の単位はラジアンです。角度を度で与える場合は、ラジアンに変換するためにπ/180を掛けます。πには次のような値を使用すると最も精度の高い計算を行うことができます。
  倍精度実数(PI#をπの値を持つ変数とします)
  PI#=3.14159265358979323846
- **数式**が倍精度実数のときは、結果も倍精度で計算した値になります。他の場合は、単精度で計算した値になります。

参照
ATN(逆正接)
COS(余弦)
SIN(正弦)

TIME$

【タイム・ダラー】

TIME
時間、時刻のこと。

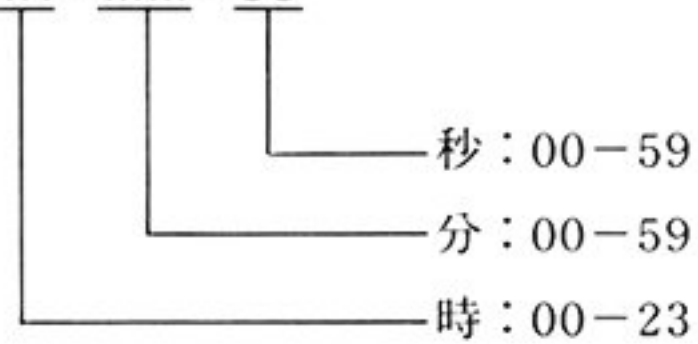
機　能　時刻を設定したり、現在の時刻を読み出したりします。

書　式
```
TIME$
TIME$="時：分：秒"
```

使用例
```
PRINT TIME$              ←現在の時刻を表示します。
TIME$="16：35：56"       ←新しく時刻を設定します。
```

解　説
- カレンダ時計に次の形式で時刻を設定します。

```
TIME$="HH：MM：SS"
```

秒：00－59
分：00－59
時：00－23

- カレンダ時計の時刻を**8文字の文字列**で読み出します。文字列の形式は時刻を設定する形式と同じです。

参　照　DATE$(日付の設定・読み出し)

T

TIME$ ON
TIME$ OFF
TIME$ STOP

【タイム・ダラー・オン】
【タイム・ダラー・オフ】
【タイム・ダラー・ストップ】

TIME
時間、時刻のこと。ここでは時間に関する割り込みの制御を行うこと。

機能 指定した時刻に発生する割り込みを可能な状態、禁止の状態、停止の状態にします。

書式

```
TIME$ |ON  |
      |OFF |
      |STOP|
```

使用例

TIME$ ON	←割り込みを可能な状態にします。
TIME$ OFF	←割り込みを禁止の状態にします。
TIME$ STOP	←割り込みを停止の状態にします。

解説

- 指定した時刻に発生する割り込みの制御を行います。
- **TIME$ ON**は割り込みを可能な状態にします。この命令を実行後は指定した時刻になると割り込みが発生し、ON TIME$ GOSUB 文で指定した割り込み処理ルーチンを実行します。
- **TIME$ OFF** は割り込みを禁止します。この命令を実行後は、指定した時刻になっても割り込みは発生しません。プログラム終了時には必ず、TIME$ OFF を実行します。
- **TIME$ STOP** は割り込みを停止の状態にします。この命令実行後は、指定した時刻になっても、そのことを記憶するだけで割り込みは発生しません。その後、TIME$ ON を実行すると割り込み処理ルーチンを実行します。

参照 ON TIME$ GOSUB(時刻による割り込み処理ルーチンの定義)
TIME$(時刻の設定)

TRON
TROFF

【トレース・オン】
【トレース・オフ】

TRace
追跡するという意味。プログラムの実行状態を確認します。

機能　プログラムの実行状態の追跡(トレース)を、設定または解除します。設定すると実行した行番号を画面に表示します。

書式　TRON
TROFF

使用例
TRON　←トレースの設定
TROFF　←トレースの解除

解説
• TRON コマンドを実行すると、実行したプログラムの行番号を角かっこ [ ] で囲んで表示していきます。
• TROFF コマンドを実行する(または NEW コマンドを実行する)と通常の状態に戻ります。

プログラム例

```
100 ' TRON / TROFF
110 ' ---  トレースするプログラム  ---
120 FOR I=1 TO 3
130   FOR J=1 TO 3
140     PRINT I;"*";J;"=";I*J
150   NEXT J
160 NEXT I
170 END

TRON
OK
RUN
[100][110][120][130][140] 1 * 1 = 1
[150][140] 1 * 2 = 2
[150][140] 1 * 3 = 3
[150][160][130][140] 2 * 1 = 2
[150][140] 2 * 2 = 4
[150][140] 2 * 3 = 6
[150][160][130][140] 3 * 1 = 3
[150][140] 3 * 2 = 6
[150][140] 3 * 3 = 9
[150][160][170]
OK
```

USR

【ユーザー】

USeR
利用者の意味。ここではユーザー独自の機械語サブルーチンのこと。

機　能　機械語で作ったサブルーチンを実行します。

書　式　USR[番号]（式）

使用例　A=USR1(P)　　←機械語サブルーチンを実行します。

解　説

- 実行する機械語サブルーチン(ユーザー関数)は、あらかじめプログラム中に準備していなければなりません。また、DEF USR文で、その実行開始アドレスをして設定する必要があります。
- **番号**にはDEF USR文で定義された番号に対応する、0～9の数字を指定します。**番号**を省略した場合には、0を指定したことになります。
- **式**を指定して、BASICから機械語サブルーチンに値を引き渡すことができます。
- USR関数の実行の方法は次のようになります。

  DEF SEG＝セグメントアドレス(実行セグメント)
  DEF USRn＝オフセットアドレス(実行開始番地)
  変数名＝USRn(数値または文字列)
  　　　　　↓　　　　↓
  　　　番号(0～9)　　式

参　照

BLOAD(機械語プログラムをメモリに読み込む)
CALL(機械語サブルーチンを呼び出す)
CLEAR(データの消去)
DEF USR(機械語サブルーチンの開始アドレスの指定)
日本語Disk BASICユーザーズマニュアル　第10章　機械語プログラム

U

VAL

【バリュー】

VALue
数値の意味。ここでは文字列を数値に変換すること。

機　能　数字で表した文字列を数値に変換します。

書　式　VAL(文字列)

使用例
```
100　A$=HEX$(1234)
110　B=VAL("&H"+A$)　　←16進表記の文字列を数値に変換します。
```

解　説
- 数値を表す**文字列**を数値に変換します。
- **文字列**の先頭が+、−、&、0から9までの数字でないときは0になります。
- **文字列**中に数値に変換できない文字がある場合は、その直前までを数値に変換します。
- 10進表記中の空白は無視します。
- 変換例は次のとおりです。

先頭の文字	解釈	文字列の例	VAL関数の値
0から9までの数字	10進表記の文字列	1 2 3 4	1 2 3 4
+	10進表記の文字列	+1 2 3 4	1 2 3 4
−	10進表記の文字列	−1 2 3 4	−1 2 3 4
&	8進表記の文字列	&1 2 3 4	6 6 8
&O または&o	8進表記の文字列	&O 1 2 3 4	6 6 8
&H または&h	16進表記の文字列	&H 1 2 3 4	4 6 6 0

参　照
HEX$(数値を16進表記の文字列に変換)
OCT$(数値を8進表記の文字列に変換)
STR$(数値を10進表記の文字列に変換)

プログラム例

```
100 ' VAL
110 ' --- 15秒ごとにBEEPを鳴らす ---
120 CLS
130 LOCATE 30,8 : PRINT TIME$
140 SS$=RIGHT$(TIME$,2)
150 S=VAL(SS$)
160 IF S MOD 15 >0 GOTO 130
170 BEEP : GOTO 130
```

VARPTR

【ヴァリアブル・ポインタ】

VARiable PoinTeR
変数(variable)を指し示す(pointer)ことから、変数のアドレスを示すこと。

機能　変数やファイル制御ブロック(FCB)のアドレスを示します。

書式　VARPTR(変数名 / #ファイル番号 [, 機能])

使用例
A=VARPTR(X)　←変数の格納されているアドレスを示します。
A=VARPTR(#1)　←ファイル制御ブロックのアドレスを示します。

解説
- **変数名**に指定した変数または配列変数の、データを記憶しているアドレスを返します。またファイル番号を指定した場合は、ファイル番号に対応するファイル制御ブロックのアドレスを返します。
- **機能**はアドレスのセグメントアドレスを返すのかオフセットアドレスを返すのかを指定します。

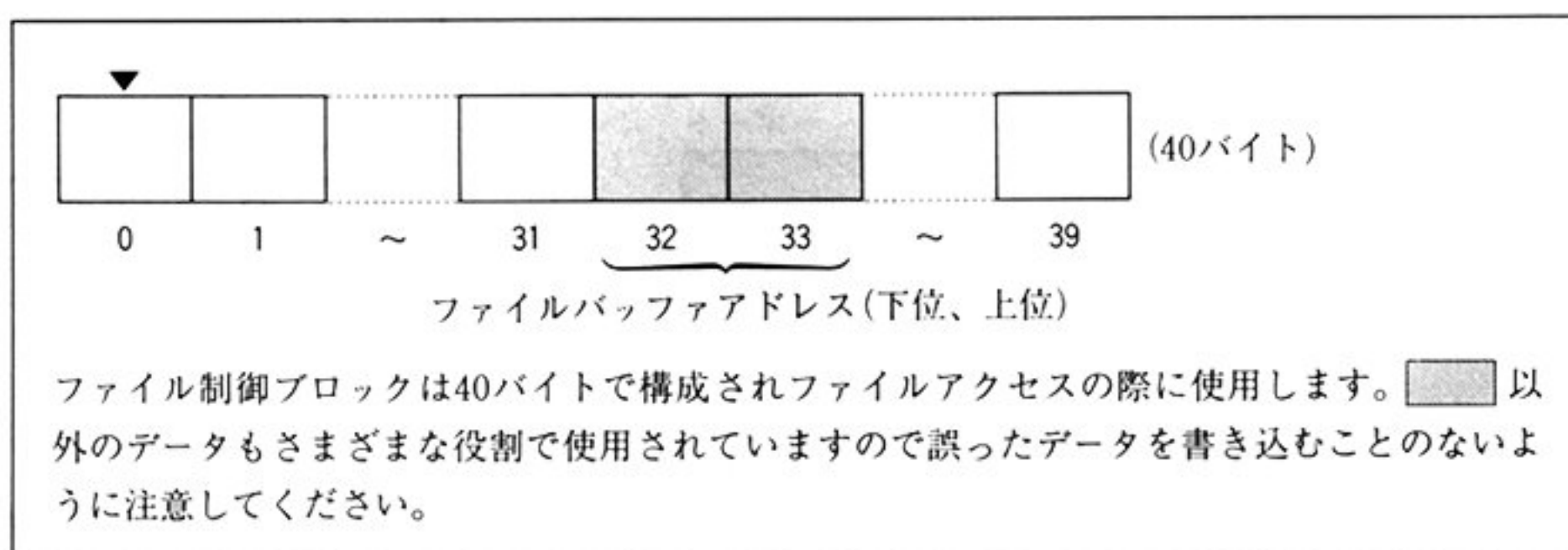
0	オフセットアドレス
1	セグメントアドレス

省略した場合は0とみなし、オフセットアドレスを返します。
- 返すアドレスは0から&HFFFFの範囲です。ただし、整数として得られる値は−32768から32767ですからVARPTR関数で得られた値に加減算を行う場合は注意してください。

• VARPTR関数で得られたアドレスはそれぞれ次の場所を指定します。

変数

▼：VARPTR 関数で得るオフセットアドレスの示す位置

ファイル制御ブロック

▼：VARPTR 関数で得るオフセットアドレスの示す位置

参　照　日本語 Disk BASIC ユーザーズマニュアル　第11章　データの内部構造

プログラム例

```
100 ' VARPTR
110 '---  配列のデータのオフセットアド
レスを調べる  ---
130 CLEAR : DIM X(10)
140 FOR I=1 TO 10
150   X(I)=INT(RND*100)
160 NEXT I
170 FOR I=1 TO 10
180   OF=VARPTR(X(I))
190   PRINT I;"offset :";HEX$(OF)
200 NEXT I
210 END
```

```
RUN
 1 offset :4
 2 offset :8
 3 offset :C
 4 offset :10
 5 offset :14
 6 offset :18
 7 offset :1C
 8 offset :20
 9 offset :24
 10 offset :28
OK
```

VIEW

【ビュー】

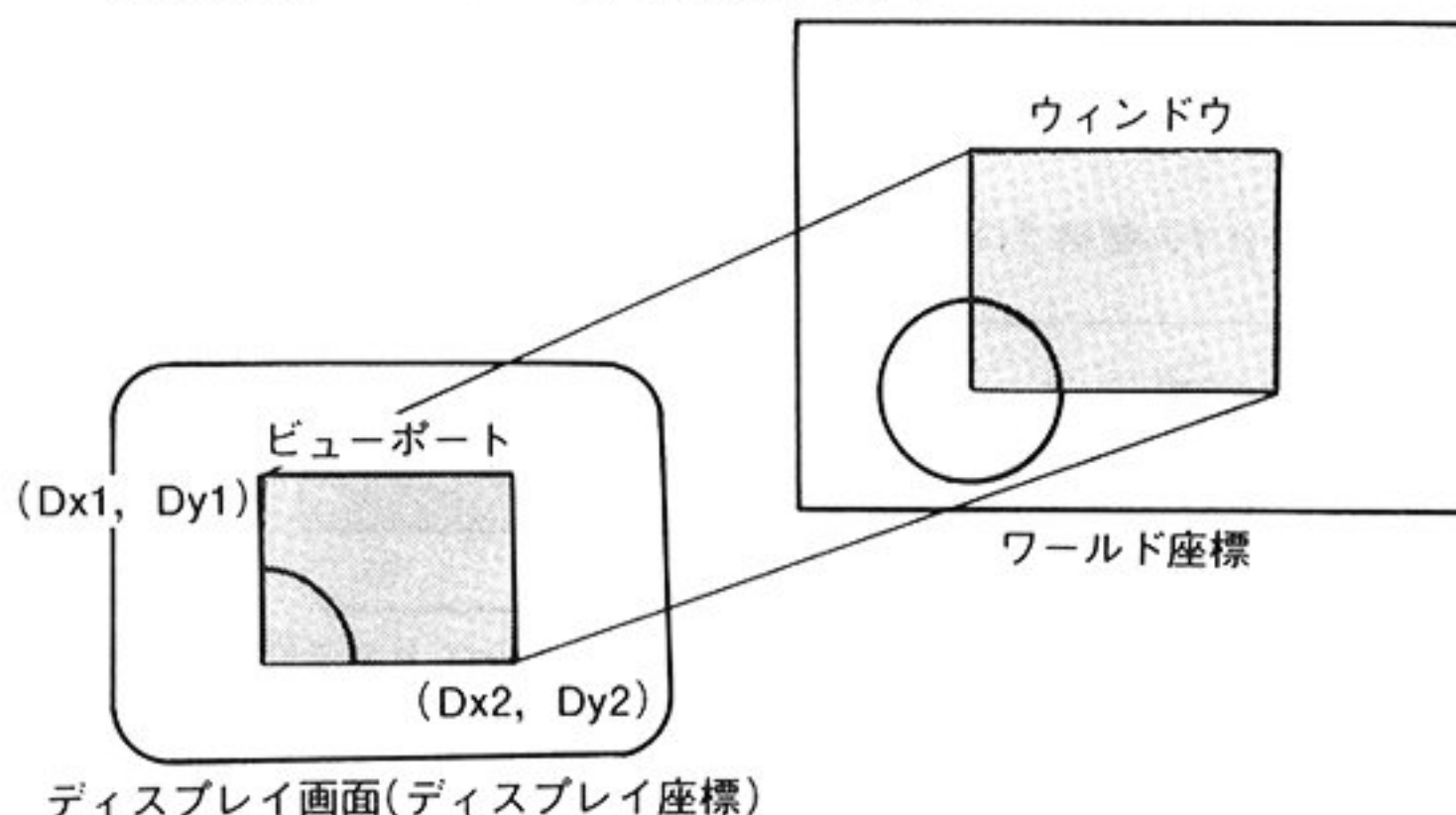
VIEW
視界、視野の意味。これから画面にグラフィックの表示範囲(ビューポート)を設定すること。

機能 画面にグラフィックの表示範囲(ビューポート)を設定します。

書式 VIEW (Dx1, Dy1)ー(Dx2, Dy2) [, [領域色] [, [境界色]]]

使用例 VIEW(100, 50)ー(500, 150),, 7 ←画面上に指定座標を対角線とする表示範囲を設定します。

解説

・ディスプレイ座標系の2点(Dx1, Dy2), (Dx2, Dy2)を対角とする四角形の範囲をグラフィック表示領域(ビューポート)に設定します。

・ビューポートにはワールド座標の中でWINDOW文によって指定したウィンドウ領域を表示します。ビューポートはグラフィックの表示範囲を限定するものですから、ビューポートの外に図形を表示することはできません。
(図形を描くのはあくまでもワールド座標に対してであり、WINDOW文やVIEW文はあくまでも表示する範囲を設定するものです。)

・**領域色**には、ビューポート設定時にビューポートを塗りつぶす色を、パレット番号で指定します。省略するとパレット番号0の色で塗りつぶします。

・**境界色**を指定すると、ビューポートの外側を指定したパレット番号の色で境界線を描きます。省略時には境界線を描きません。

・ディスプレイ座標系の座標は Dx1＜Dx2, Dy1＜Dy2 でなければなりません。またディスプレイ座標より大きい値を指定した場合は Illegal function call(違法関数呼び出し)エラーになります。

- 一度設定したビューポートは、再度 VIEW 文を実行しないかぎり変化しません。また、SCREEN 文により初期化されます。
- VIEW 文をいくつか使用すると、画面上の別々の位置に、別々の大きさで図形を表示することができます。
- 最終参照座標はビューポートの左上隅の点、すなわち、スクリーン座標系の(0，0)の点に移動します。

参 照

CLS(画面の消去)
SCREEN(グラフィック画面のモードを設定)
WINDOW(ワールド座標の表示範囲の設定)
日本語 Disk BASIC ユーザーズマニュアル　第6章　テキスト画面とグラフィック画面

VIEW

【ビュー】

VIEW
視界、視野の意味。ここではビューポートの位置を知ること。

〔機　能〕　ビューポートの位置を示します。

〔書　式〕　VIEW(機能)

〔使用例〕　SX1=VIEW(0)　　←ビューポートの左上隅のX座標を与えます。

〔解　説〕
- ビューポートの位置を、**機能**で指定したディスプレイ座標の座標値で返します。
- **機能**は0から3までの整数を指定し、意味は次のとおりです。

機能	意　　味
0	ビューポートの左上隅の点のX座標
1	ビューポートの左上隅の点のY座標
2	ビューポートの右下隅の点のX座標
3	ビューポートの右下隅の点のY座標

〔参　照〕　VIEW(ビューポートの設定)
日本語Disk BASICユーザーズマニュアル　第6章　テキスト画面とグラフィック画面

WAIT

【ウェイト】

WAIT
待つことから、プログラムの実行を指定値になるまで待たせること。

機能　I/O ポートのデータを調べ、指定値になるまで待ちます。

書式　WAIT　ポート番号, データ [, パターン]

解説

- WAIT 文は**ポート番号**で指定した I/O ポートのデータが指定のビットパターンになるまでプログラムの実行を停止します。
- 読み出したデータは**パターン**で指定したデータと XOR 演算を行い、さらに**データ**と AND 演算を行います。
- 演算の結果が 0 であれば再び I/O ポートのデータを読み出します。0 でなければ次の命令を実行します。
- **パターン**を省略すると 0 を指定したことになります。
- WAIT 文により無限ループに入ってしまうことがあります。ループから抜け出すためには、ウォームスタート(STOP キーを押しながらリセットボタンを押す)を行ってください。

W

WHILE～WEND

【ホァイル～エンド】

WHILE ～ While END
～の間、進むまたは行くという意味から、ある条件のもとで繰り返し処理を行うこと。

機　能　WHILE 直後の条件が満たされている間、WHILE～WEND の間にある一連の命令を実行します。

書　式

```
WHILE　条件式
    :
WEND
```

使用例

```
100  I=1
110  WHILE I<60          ← I <60のあいだ WEND までの命令を実行します。
        :
300     I=I+2
310  WEND
```

解　説

- WHILE 文と WEND 文は必ず組み合わせて用い、処理の繰り返し(ループ)を形成します。
- **条件式**は WHILE～WEND ループを実行するかどうかの判定するための式です。
- **条件式**を評価して、その結果が真(−1)であれば、WHILE～WEND 間の命令を実行します。実行後再び、**条件式**を評価します。この繰り返しを**条件式**の結果が偽(0)になるまで繰り返します。偽になると WEND 文の次の命令を実行します。したがって、**条件式**が初めから偽であれば、WHILE～WEND 間の命令は一度も実行しません。
- WHILE～WEND 文は、FOR～NEXT 文と同様に入れ子構造にすることができます。それぞれの WEND 文は、直前に実行した WHILE 文に対応付けますが、WHILE 文と WEND 文の対応が付かなければ WHILE without WEND(WHILE があるのに WEND がない)または WEND without WHILE(WEND があるのに WHILE がない)エラーになります。

参　照　FOR～NEXT(繰り返し実行)

プログラム例

```
100 ' WHILE - WEND
110 ' ---  指定数まで積算する  ---
120 INPUT "積算する数は ";N
130 I=0:S=0
140 WHILE I=<N
150   S=S+I
160   I=I+1
170 WEND
180 PRINT "答えは          ";S
190 END
```

```
RUN
積算する数は ? 100
答えは         5050
OK
```

WIDTH

【ウィドゥス】

WIDTH
広さ、幅のこと。これから画面やファイルの大きさを設定すること。

機能 画面に表示する文字数や各種ファイルの1行の長さを設定します。

書式

WIDTH　桁数 [, 行数]
WIDTH "デバイス名：", 文字数
WIDTH #ファイル番号, 文字数

使用例

WIDTH 80, 25　　←画面表示を80桁25行に設定します。

解説

WIDTH　桁数 [, 行数]

- テキスト画面に表示する文字の**桁数**と**行数**を設定します。指定できる値は次のとおりです。

桁数	40または80
行数	20または25。省略時は現在の行数のまま変化しない

WIDTH "デバイス名：", 文字数

- **プリンタ(LPT1：), 通信回線(COM：)**の1行の長さ(文字数)を設定します。ここで設定した文字数ごとに、自動的にCR/LFコードを出力します。文字数には0から255までの整数を指定し、0は256とみなします。ただし255を設定したときは自動的にCR/LFコードを出力しません。BASIC起動時は255に設定されています。
- WIDTH "LPT1：" はWIDTH LPRINT文と同じ機能です。

WIDTH # ファイル番号, 文字数

- **ファイル番号**で指定したプリンタ、通信回線のファイルの1行の文字数を設定します。ここで設定した文字数ごとに、自動的にCR/LFコードを出力します。**文字数**には0から255までの整数を指定し、0は256とみなします。ただし255を設定したときは自動的にCR/LFコードを出力しません。BASIC起動時は255に設定されています。

参照 WIDTH LPRINT(プリンタの1行の長さの設定)

WIDTH LPRINT

【ウィドゥス・エル・プリント】

WIDTH Line PRINT
line はここではプリンタのこと。プリンタの1行の文字数を設定すること。

機　能　プリンタが印字する1行の文字数を設定します。

書　式　WIDTH LPRINT　文字数

使用例　WIDTH LPRINT 80　　←1行の文字数を80に設定します。

解　説
- プリンタが印字する1行の**文字数**を0から255の範囲で指定します。0を指定したときは256になります。
- 指定した**文字数**を超えると自動的にCR/LFコードをプリンタに出力します。ただし255を指定したときは、自動的にCR/LFコードを出力しません。
- **文字数**は1バイト文字を1文字として数えたものです。
- BASIC起動時は255(自動的にCR/LFコードを出力しない)に設定されます。

参　照　WIDTH(ファイルの1行の文字数を設定)

プログラム例

```
100 ' WIDTH LPRINT
110 ' ---  プリンタに送る1行を40桁に設定する  ---
120 WIDTH LPRINT 40
130 FOR I=1 TO 160
140   LPRINT "*";
150 NEXT I
160 END

RUN
****************************************
****************************************
****************************************
****************************************
```

WINDOW

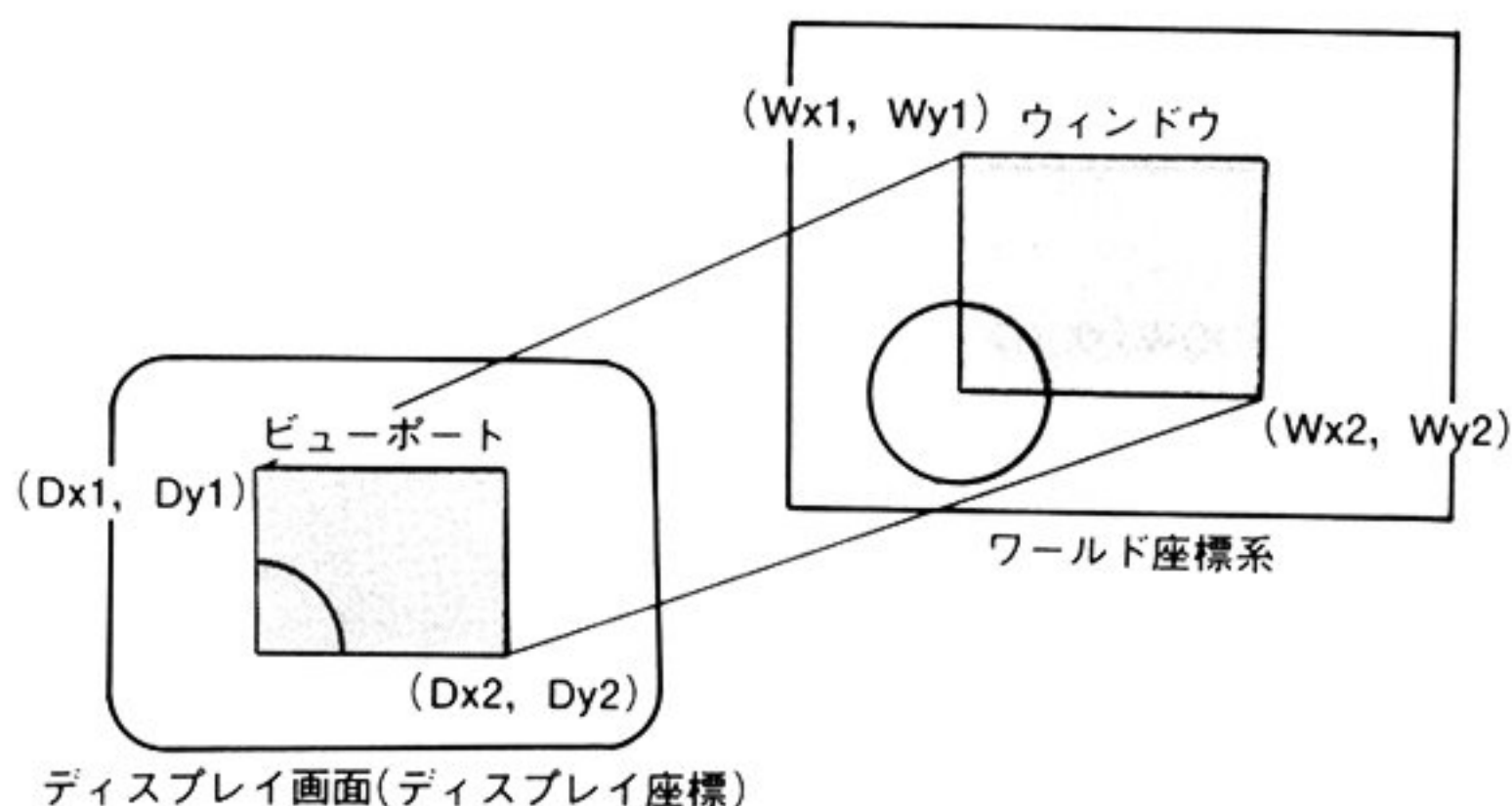
【ウィンドウ】

WINDOW
窓のこと。外の風景を窓を通して見ると、景色が限定されることから。

機能　画面に表示するワールド座標系の領域(ウィンドウ)を設定します。

書式　*WINDOW (Wx1, Wy1)−(Wx2, Wy2)*

使用例　WINDOW(0, 0)−(1256, 1258)　　←ウィンドウの大きさを変えます。

解説　• ワールド座標系の座標(**Wx1, Wy1**)と(**Wx2, Wy2**)を対角とする領域をウィンドウに設定します。ここで設定したウィンドウ内のグラフィックは画面上のビューポートを通して見ることができます。ただし指定する座標は **Wx1<Wx2**、**Wy1<Wy2** でなければなりません。

• 一度設定したウィンドウは、再度 WINDOW 文または SCREEN 文を実行しない限り変化しません。また WINDOW 文の実行により最終参照座標はウィンドウの左上隅の点に移動します。

参照　VIEW(ビューポートの設定)
WINDOW(ウィンドウの設定位置)

WINDOW

【ウィンドウ】

WINDOW
窓のこと。ここではウィンドウの位置を知ること。

機　能　ウィンドウの位置をワールド座標の座標値で知らせます。

書　式　WINDOW(機能)

使用例　WX1=WINDOW(0)　←ウィンドウの左上隅のX座標を知らせます。

解　説

- ウィンドウの位置を、**機能**で指定したワールド座標系の座標値で返します。
- **機能**は0から3までの数値で、意味は次のとおりです。

機能	意　味
0	ウインドウの左上隅の点のX座標
1	ウィンドウの左上隅の点のY座標
2	ウィンドウの右下隅の点のX座標
3	ウィンドウの右下隅の点のY座標

参　照　WINDOW(ウィンドウの設定)
日本語Disk BASICユーザーズマニュアル　第6章　テキスト画面とグラフィック画面

WRITE

【ライト】

WRITE
書くという意味。画面に文字を表示すること。

機　能　データを画面に表示します。

書　式　WRITE　式 [, 式] ・・・

使用例　WRITE 123, 456, "ABC"　　←画面に123, 456, "ABC"と表示します。

解　説
- 式で指定した数値データや文字データを画面に表示します。
- 式を２つ以上続けるときはカンマ(,)あるはセミコロン(；)で区切って並べていきますが、両者に違いはありません。
- WRITE 文は PRINT 文と異なりデータとデータの間をカンマ(,)で区切って表示していきます。また文字データは二重引用符(")で囲んで表示します。

参　照　PRINT(画面にデータを表示する)
WRITE#(シーケンシャルファイルにデータを書き込む)

プログラム例

```
100 ' WRITE
110 ' ---  WRITE と PRINT の違い  ---
120 A=12.3456:B=1.23E+07:C$="EPSON":D$="COMPUTER"
130 PRINT "WRITE--->";:WRITE A,B,C$,D$
140 PRINT "PRINT--->";:PRINT A;B;C$;D$
150 END

RUN
WRITE--->12.3456,1.23E+07,"EPSON","COMPUTER"
PRINT---> 12.3456  1.23E+07 EPSONCOMPUTER
OK
```

WRITE＃

【ライト・シャープ】

WRITE
書くという意味。ここではファイルにデータを書き込むこと。

機能 データをシーケンシャルファイルに書き込みます。

書式 WRITE　＃ファイル番号，式［，式］・・・

使用例

```
100 OPEN "TEST" FOR OUTPUT AS ＃1
200 A$= "ABC" : B$= "DEF"
300 WRITE＃1, A$, B$
```

←シーケンシャルファイル TEST に "ABC", "DEF" と書き込みます。

解説

- **ファイル番号**で指定したシーケンシャルファイルに、式のデータを書き込みます。
- WRITE＃ 文は、WRITE 文と同じように、データとデータの間に自動的にカンマ(,)を挿入し、文字データを二重引用符で囲みます。したがって、PRINT＃ 文のようにプログラムで区切りデータを挿入する必要がありません。

A$="Tanaka"

B$="123−456"

WRITE＃1, A$, B$

はファイルに次のような内容を書き込みます。

"	T	a	n	a	k	a	"	,	"	1	2	3	−	4	5	6	"	CR	LF

CR：CR コード

LF：LF コード

- PRINT＃ 文は正の数値データの前に空白を出力しますが、WRITE＃ 文は出力しません。
- 最後の式の後ろには CR/LF コードを書き込みます。

参照 PRINT＃(シーケンシャルファイルにデータを書き込む)
WRITE(画面にデータを表示)

プログラム例

```
100 ' WRITE#
110 ' ---  シーケンシャルファイルにデータを書き込む  ---
120 OPEN "2:DATA2" FOR OUTPUT AS #1
130 FOR I=1 TO 3
140   READ NM$,TEL$
150   WRITE #1,NM$,TEL$
160   '-- PRINT # と違いデータをカンマで区切る必要はない --
170 NEXT I
180 CLOSE #1
190 END
200 DATA Tanaka,123-4567
210 DATA Sato,456-7890
220 DATA Suzuki,987-6543
```

第2章

エラーメッセージ

BASIC実行時に発生するエラーメッセージには、次のようなものがあります。表の「エラーメッセージ」には1段目に表示されるエラーメッセージ、2段目に日本語での意味、3段目にエラーコードを示します。

エラーメッセージ	エラーメッセージの内容(発生の原因と対策)
/0 ゼロで除算 11	数値を0で割ったとき/0を表示し、計算結果として割られる数値の符号を持つ数値の上限値を与えてプログラムの実行を継続します。
Bad allocation table FATに異常を発見 69	ディスクのFATが何らかの原因で破壊されています。
Bad drive number ドライブ番号の誤り 70	接続していない周辺機器のドライブ番号やデバイス名を指定しています。
Bad file name ファイル名が間違っている 56	デバイス名としてBASICが扱えるもの以外を指定しています。またそのデバイスで指定することのできない入出力モードを指定しています。
Bad file number ファイル番号が間違っている 52	同時にオープンできるファイルの数を超えたファイル番号を指定しています。
Bad track/sector トラック番号、セクタ番号が間違っている 71	DSKI$、DSKO$命令で指定したトラック、セクタ番号がディスクの仕様外の値です。
Can't continue プログラムの継続不可能 17	CONTコマンドでプログラムの実行を継続しようとしたが、停止中にプログラムの修正を行ったため継続できません。
Deleted record レコード消去済み 72	プログラムの実行では発生しません。ERRORコマンドでのみ表示することができます。
Direct statment in file ファイル中に直接命令が存在 57	アスキーセーブしたプログラムファイル中に行番号のない行が存在します。このファイルはプログラムファイルでない可能性もあります。

エラーメッセージ	エラーメッセージの内容(発生の原因と対策)
Disk full ディスクが満杯 68	ファイルの数がディレクトリで管理できる数を超えたか、ディスク上にデータを記録する領域がありません。
Disk I/O error ディスクの入出力エラー 64	ディスク入出力中にエラーが発生しました。致命的なエラーであり回復することができません。
Disk offline ディスクが使用不可能 62	フロッピーディスクを正しくドライブにセットしていません。 ハードディスク上に BASIC 領域がありません。
Duplicate definition 二重定義 10	配列を二重に定義しています。
Division by Zero ゼロで除算 11	整数の除算(¥、MOD)の実行時にゼロで除算を行っています。Division by Zero を表示し実行を中断します。
Duplicate label ラベル重複 31	同一プログラム中に同じラベル名を定義しています。
Feature not available 指定した機能は実行不可能 33	拡張機能を指定したときに使用できる命令を実行しようとしたり、拡張ボード内の命令をボードを未装着の状態で使用しています。
FIELD overflow FIELD 文の指定オーバー 50	FIELD 文で指定できるランダムファイルバッファの大きさは256バイト以下です。
File already exist ファイルが既に存在 65	NAME コマンドで変更するファイル名を持つファイルは既に存在します。
File already open ファイルはすでにオープン済み 54	すでにオープンしているファイルに対して OPEN 文または KILL 文を実行しました。
File not found ファイルが見つからない 53	OPEN、NAME 文で指定したファイルがありません。

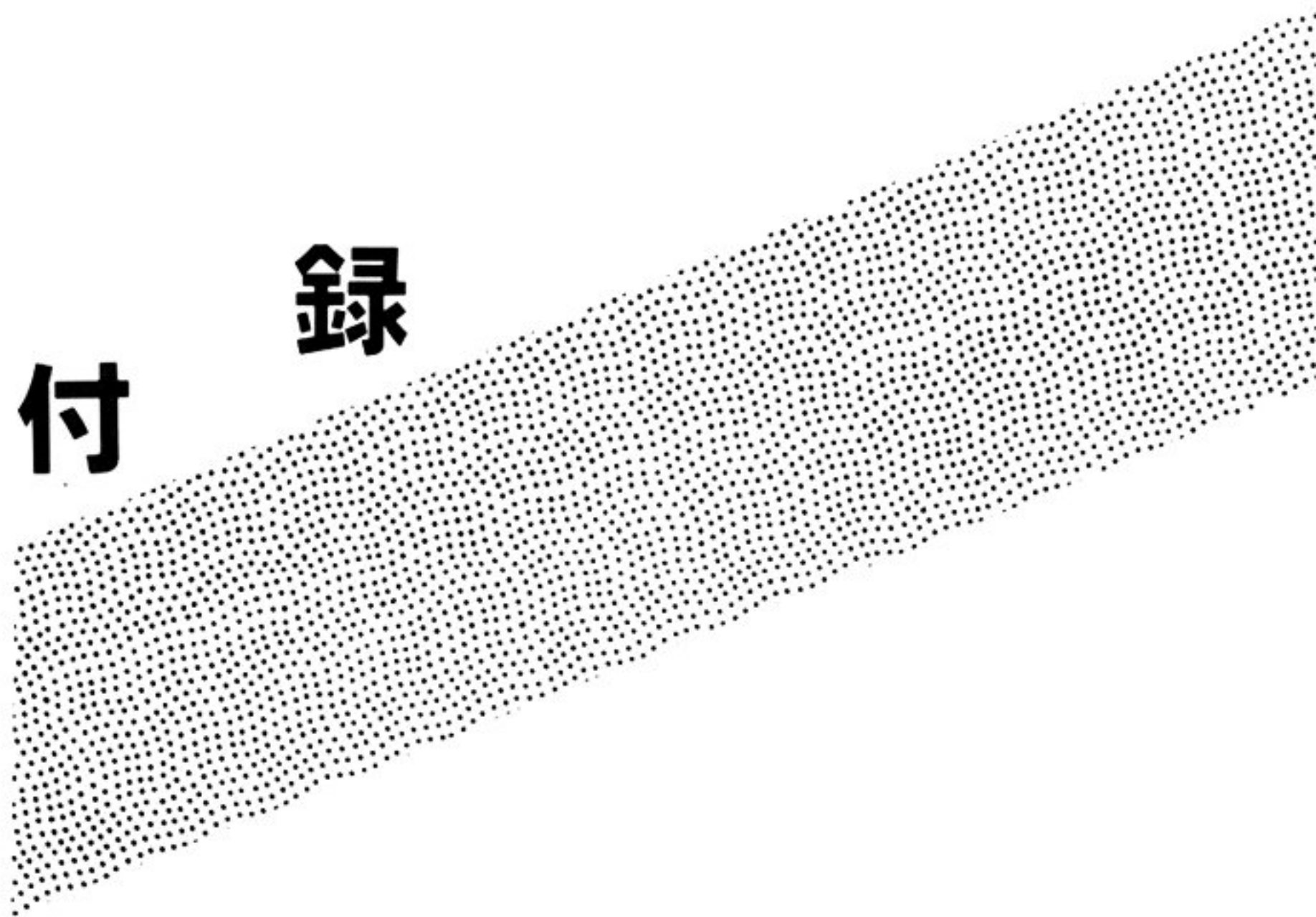
エラーメッセージ	エラーメッセージの内容(発生の原因と対策)
File not open ファイルをオープンしていない 60	オープンしていないファイルに対してINPUT#やPRINT#文などの入出力命令を実行しています。
File write protected ファイルが書き込み禁止 61	指定したファイルにSET文で書き込み禁止の設定がしてあります。 フロッピーディスクにライトプロテクトタブのような書き込み禁止の処理がしてあります。
FOR without NEXT NEXT文のないFOR文 26	FOR文に対応するNEXT文がないためFOR〜NEXTを正しく実行できません。NEXT文よりFOR文の数が多くなっています。
Illegal direct 違法なダイレクトモードでの実行 12	DEF FN文のようなダイレクトモードで実行できない命令をダイレクトモードで入力した。
Illegal function call 違法関数呼び出し 5	指定可能範囲外のパラメータを関数に与えて実行しようとしています。
Illegal operation 間違った操作 74	誤ったデータの入力を行っています。
Input past end 読み込むデータは終了 55	データのないファイルにINPUT#文などの入力命令を実行したり、すべてのデータを入力し終えたシーケンシャルファイルに入力命令を実行しています。ファイルの終わりはEOF関数などで調べることができます。
Internal error 内部エラー 51	BASICの内部処理の際に不正なコマンドが発生しています。一般には起こり得ないエラーです。
Line buffer overflow 1行の桁数あふれ 23	通信回線からデータ受信を行っている際に受信バッファがオーバーフローしました。 プログラムを内部コードに変換する際に内部コードが512バイトを超えました。
Missing operand オペランドがない 22	命令や関数を実行する際に必要なパラメータを指定していません。 または、演算子の右側に何も記述がありません。
NEXT without FOR FOR文のないNEXT文 1	FOR文を実行せずにNEXT文を実行しようとしています。FOR文よりNEXT文の数が多くなっています。

エラーメッセージ	エラーメッセージの内容(発生の原因と対策)
No RESUME RESUME 文がない 19	ON ERROR GOTO 文で分岐した処理が RESUME 文なし(エラ復旧なし)で終了した場合に発生します。
Out of DATA データがない 4	READ 文で読むデータがありません。READ 文で読み込むデータの数と DATA 文で定義したデータの数を比べてください。
Out of memory メモリ容量不足 7	プログラムを実行するためのメモリ容量が足りません。または GOSUB 文や FOR〜NEXT 文のネスト(入れ子)が深すぎます。
Out of string space 文字列領域の不足 14	文字列を格納するためのメモリ容量が足りません。
OV 桁あふれ 6	演算結果や入力した数値の絶対値が大きすぎるとき OV を表示し、可能な限り大きな値で継続します。
Overflow 桁あふれ 6	整数の除算や整数型変数への代入などで扱う数値が大きすぎます。この場合は Overflow を表示し実行を中断します。
Rename across disks ディスク指定の間違い 73	NAME コマンドで指定したファイルに異なったディスク番号を指定しています。
RESUME without error エラーなしの RESUME 文 20	ON ERROR GOTO 文によるエラー処理ルーチンを実行せずに RESUME 文を実行しようとしています。
RETURN without GOSUB GOSUB 文のない RETURN 文 3	GOSUB 文のようなサブルーチンを実行していないのに RETURN 文を実行しようとしています。
Sequential after PUT PUT 後のシーケンシャル出力 58	プログラムの実行では発生しません。ERROR コマンドでのみ表示することができます。
Sequential I/O only シーケンシャル入出力のみ 59	アスキーセーブしていないファイルを MERGE 文または MERGE オプション付きの CHAIN 文でロードしようとしています。

エラーメッセージ	エラーメッセージの内容(発生の原因と対策)
String formula too complex 文字式が複雑すぎる 16	計算式や文字列が複雑すぎます。FN 関数や文字列関数のカッコの数を減らし、中間結果を変数に代入するようにしてください。
String too long 文字列が長すぎる 15	"　"で囲まれた文字列の長さが255バイトを超えているか、文字列の加算の結果が255バイトを超えています。
Subscript out of range 添字が範囲外 9	配列変数で指定した添字が DIM 文、OPTION BASE 文で指定した範囲を超えている。
Syntax error 文法の間違い 2	プログラムの記述が文法に従っていません。 「)」の数が「(」よりも多いときにも、このエラーが起ります。
Type mismatch 型の不一致 13	式の右辺と左辺、あるいは関数の引数などに型の違う変数や値を指定しています。 文字型と数値型との間違いがあります。
Undefined label ラベルがない 32	参照したラベルがプログラム中に定義してありません。
Undefined line number 行番号がない 8	参照した行番号がプログラム中に存在しません。
Undefined user function ユーザー関数の未定義 18	DEF FN 文で定義していない FN 関数か DEF USR 関数で定義していない USR 関数を実行しようとしています。変数名の先頭に FN を使用するとユーザー定義関数の実行とみなされます。
Unprintable error エラーメッセージ表示不可能 21	ERROR 文でエラーメッセージの定義していないエラーコードを指定しています。
WEND without WHILE WHILE 文のない WEND 文 30	WHILE 文を実行せずに WEND 文を実行しようとしました。WHILE 文より WEND 文の数が多くなっています。
WHILE without WEND WEND 文のない WHILE 文 29	WHILE 文に対応する WEND 文がないため WHILE～WEND を正しく実行できません。WEND 文より WHILE 文の数が多くなっています。

エラーコード	メッセージ
1	NEXT without FOR
2	Syntax error
3	RETURN without GOSUB
4	Out of DATA
5	Illegal function call
6	OV または Overflow
7	Out of memory
8	Undefined line number
9	Subscript out of range
10	Duplicate definition
11	/0 または Division by zero
12	Illegal direct
13	Type mismatch
14	Out of string space
15	String too long
16	String formula too complex
17	Can't continue
18	Undefined user function
19	No RESUME
20	RESUME without error
21	Unprintable error
22	Missing operand
23	Line buffer overflow
26	FOR without NEXT
29	WHILE without WEND
30	WEND without WHILE

エラーコード	メッセージ
31	Duplicate label
32	Undefined label
33	Feature not available
50	FIELD overflow
51	Internal error
52	Bad file number
53	File not found
54	File already open
55	Input past end
56	Bad file name
57	Direct statment in file
58	Sequential after PUT
59	Sequential I/O only
60	File not open
61	File write protected
62	Disk offline
64	Disk I/O error
65	File already exist
68	Disk full
69	Bad allocation table
70	Bad drive number
71	Bad track/sector
72	Deleted record
73	Rename across disks
74	Illegal operation

付録

A 文字コード表

下位ビット＼上位ビット	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0			スペース	0	@	P	`	p	▁	┴		ー	タ	ミ	═	╳
1	HELP		!	1	A	Q	a	q	▂	┬	。	ア	チ	ム	╞	円
2			"	2	B	R	b	r	▃	┤	「	イ	ツ	メ	╪	年
3			#	3	C	S	c	s	▄	├	」	ウ	テ	モ	╡	月
4			$	4	D	T	d	t	▅	▔	、	エ	ト	ヤ	◢	日
5			%	5	E	U	e	u	▆	─	・	オ	ナ	ユ	◥	時
6			&	6	F	V	f	v	▇	│	ヲ	カ	ニ	ヨ	◣	分
7	BEEP		'	7	G	W	g	w	█	▕	ァ	キ	ヌ	ラ	◤	秒
8	BS		(	8	H	X	h	x	▏	┌	ィ	ク	ネ	リ	♠	
9	TAB		)	9	I	Y	i	y	▎	┐	ゥ	ケ	ノ	ル	♥	
A	LF		*	:	J	Z	j	z	▍	└	ェ	コ	ハ	レ	♦	
B	HOME	ESC	+	;	K	[	k	{	▌	┘	ォ	サ	ヒ	ロ	♣	
C	CLR	→	,	<	L	¥	l	¦	▋	╭	ャ	シ	フ	ワ	●	＼
D	↵	←	-	=	M	]	m	}	▊	╮	ュ	ス	ヘ	ン	○	
E		↑	.	>	N	^	n	~	▉	╰	ョ	セ	ホ	゛	／	
F		↓	/	?	O	_	o		┼	╯	ッ	ソ	マ	゜	＼	

B 予約語

BASICではコマンド、ステートメント、関数、演算子など特定目的の使用が決められている単語を予約語としています。あらかじめ使用を決めているのは、それらの単語を変数として使用したときに起きる混乱を避けるためです。したがって、予約語を変数名として使用することはできません。

次にあげる語がBASICの予約語です。これらを変数名、ラベル名として使用することはできません。

A
ABS
AKCNV$
AND
ASC
ATN
ATTR$
AUTO

B
BEEP
BLOAD
BSAVE

C
CALL
CDBL
CHAIN
CHR$
CINT
CIRCLE
CLEAR
CLOSE
CLS
CMD
COLOR
COM
COMMON
CONSOLE
CONT
COPY
COS
CSNG
CSRLIN
CVD
CVI
CVS

D
DATA
DATE$
DEF
DEFDBL
DEFINT
DEFSNG
DEFSTR
DELETE
DIM
DRAW
DSKF
DSKI$
DSKO$

E
EDIT
ELSE
END
EOF
EQV
ERASE
ERL
ERR
ERROR
EXP

F
FIELD
FILES
FIX
FN
FOR
FPOS
FRE

G
GET
GOTO
GO TO
GOSUB

H
HELP
HEX$

I
IEEE
IF
IMP
INKEY$
INP
INPUT
INPUT$
INSTR
INT
IRESET
ISET

J
JIS$

K
KACNV$
KANJI
KEXT$
KEY
KILL
KINPUT
KINSTR
KLEN
KMID$
KNJ$
KPLOAD
KTYPE

L
LEFT$
LEN
LET
LFILES
LINE
LIST
LLIST
LOAD
LOC
LOCATE
LOF
LOG
LPOS
LPRINT
LSET

M
MAP

MERGE
MID$
MKD$
MKI$
MKS$
MOD
MON

N NAME
NEXT
NEW
NOT

O OCT$
OFF
ON
OPEN
OPTION
OR
OUT

P PAINT
PEEK
PEN
POINT
POKE
POLL
POS
PRESET
PRINT
PSET
PUT

R RANDOMIZE
RBYTE
READ
REM
RENUM
RESTORE
RESUME
RETURN
RIGHT$
RND
ROLL
RSET
RUN

S SAVE
SCREEN
SEARCH
SEG
SET
SGN
SIN
SPACE$
SPC
SQR
SRQ
STATUS
STEP
STOP
STR$
STRING$
SWAP

T TAB
TAN
THEN
TIME$
TO
TROFF
TRON

U USING
USR

V VAL
VARPTR
VIEW

W WAIT
WBYTE
WEND
WHILE
WIDTH
WINDOW
WRITE

X XOR

C 誘導関数

以下の誘導式により、組み込み関数として用意されていない関数を利用することができます。
DEF FN 文を用いて定義することもできます。

目的の数学関数	組み込み関数からの誘導式
sec x（セカント）	1/COS(X)
cosec x（コセカント）	1/SIN(X)
cot x（コタンジェント）	1/TAN(X) または COS(X)/SIN(X)
arcsin x（アーク・サイン）	ATN(X/SQR(−X*X+1))
arccos x（アーク・コサイン）	−ATN(X/SQR(−X*X+1))+PI#/2
arcsec x（アーク・セカント）	ATN(SQR(X*X−1)+(SGN(X)−1))*PI#/2
arccosec x（アーク・コセカント）	ATN(1/SQR(X*X−1))+(SGN(X)−1)*PI#/2
arccot x（アーク・コタンジェント）	−ATN(X)+PI#/2
sinh x（ハイパボリック・サイン）	(EXP(X)−EXP(−X))/2
cosh x（ハイパボリック・コサイン）	(EXP(X)+EXP(−X))/2
tanh x（ハイパボリック・タンジェント）	−EXP(−X)/(EXP(X)+EXP(−X))*2+1
sech x（ハイパボリック・セカント）	2/(EXP(X)+EXP(−X))
cosech x（ハイパボリック・コセカント）	2/(EXP(X)−EXP(−X))
coth x（ハイパボリック・コタンジェント）	EXP(−X)/(EXP(X)−EXP(−X))*2+1
arcsinh x（ハイパボリック・アーク・サイン）	LOG(X+SQR(X*X+1))
arccosh x（ハイパボリック・アーク・コサイン）	LOG(X+SQR(X*X−1))
arctanh x（ハイパボリック・アーク・タンジェント）	LOG(1+X)/(1−X)/2
arcsech x（ハイパボリック・アーク・セカント）	LOG((SQR(−X*X+1)+1)/X)
arccosech x（ハイパボリック・アーク・コセカント）	LOG((SGN(X)*SQR(X*X+1)+1)/X)
arccoth x（ハイパボリック・アーク・コタンジェント）	LOG((X+1)/(X−1))/2
$\log_{10} x$（常用対数）	LOG(X)/LOG(10#)
sin x（度）	SIN(X*PI#/180)
cos x（度）	COS(X*PI#/180)
tan x（度）	TAN(X*PI#/180)

PI#=3.14159265358979323846とします。

D 機能別索引

プログラムの作成・編集・実行

AUTO　行番号を自動的に発生 …… 10
DELETE　プログラム行の削除 …… 45
EDIT　プログラム行の修正 …… 54
LIST　プログラムリストの表示 …… 106
LLIST　プログラムリストのプリンタ出力 …… 106
MERGE　プログラムのマージ(結合) …… 115
NEW　プログラムの消去 …… 121
RENUM　行番号の変更 …… 163

CONT　プログラムの実行再開 …… 33
LOAD, R　プログラムの実行 …… 107
RUN　プログラムの実行 …… 170
TROFF　プログラムのトレースの中止 …… 193
TRON　プログラムのトレース …… 193

BLOAD　機械語プログラムのロード(読み込み) …… 12
BSAVE　機械語プログラムのセーブ(書き込み) …… 13
FILES　ファイル名の表示 …… 62
KILL　ファイルの削除 …… 90
LFILES　ファイル名のプリンタへの出力 …… 62
LOAD　プログラムのロード(読み込み) …… 107
NAME　ファイル名の変更 …… 120
SAVE　プログラムのセーブ(書き込み) …… 171

REM　注釈行の設定 …… 162
　注釈行の設定 …… 162

NEW ON　システムの再起動 …… 122

プログラムの実行の流れ

CHAIN　プログラムの連鎖(チェイン) …… 16
END　プログラムの実行終了 …… 55
FOR～NEXT　繰り返して実行 …… 64

GOSUB～RETURN　サブルーチンの実行 …… 71
GOTO　プログラムの分岐 …… 72
IF～GOTO～ELSE　条件・判断処理による分岐 …… 75
IF～THEN～ELSE　条件・判断処理による分岐 …… 75
ON～GOTO　プログラムの分岐の設定 …… 126
ON～GOSUB　サブルーチンの設定 …… 126
STOP　プログラムの実行停止 …… 184
WHILE～WEND　条件が真のときループ内を実行 …… 202

データの管理・定義

CLEAR　各変数領域の設定・変数の消去 …… 22
COMMON　変数の引き渡し …… 31
DATA　READ 文で読み込むデータの定義 …… 39
DEFINT　変数を整数型に定義 …… 42
DEFDBL　変数を倍精度実数型に定義 …… 42
DEFSNG　変数を単精度実数型に定義 …… 42
DEFSTR　変数を文字型に定義 …… 42
DIM　配列変数の定義 …… 46
ERASE　配列変数の消去 …… 57
FRE　ユーザーエリアの未使用領域を示す …… 67
LET　代入文 …… 100
OPTION BASE　配列変数の添字の下限値の設定 …… 136
READ　データの読み込み …… 161
RESTORE　READ 文で読み込む DATA 文行の設定 …… 164
SEARCH　配列変数内での指定データの位置 …… 177
SWAP　２変数間の値の交換 …… 188
VARPTR　変数・ファイル制御ブロックのアドレスを知らせる …… 196

データファイル

CLOSE　ファイルのクローズ …… 23
CVD　８バイト文字列を倍精度実数に変換 …… 38
CVI　２バイト文字列を整数に変換 …… 38
CVS　４バイト文字列を単精度実数に変換 …… 38
EOF　ファイルの終わりの判別 …… 56
FIELD　ファイルバッファへの文字変数の割り当て …… 61
FPOS　ディスク上の読み書きしたセクタを知らせる …… 66
GET#　ランダムファイルからレコードの読み出し …… 68
INPUT#　シーケンシャルファイルからデータの読み出し …… 79
INPUT$　シーケンシャルファイルからデータの読み出し …… 80
LINE INPUT#　シーケンシャルファイルから一行の読み出し …… 104
LOC　ファイルの現在位置を知る …… 108
LOF　ファイルの大きさを示す …… 110

LSET　ファイルバッファへの文字データの書き込み(左寄せ)……113
MKD$　倍精度実数を8バイト文字列に変換……118
MKI$　整数を2バイト文字列に変換……118
MKS$　単精度実数を4バイト文字列に変換……118
OPEN　ファイルのオープン……134
PRINT#　シーケンシャルファイルへのデータの書き込み……154
PRINT# USING　書式指定によるシーケンシャルファイルへのデータの書き込み……154
PUT#　ランダムファイルへのレコードの書き込み……157
RSET　ファイルバッファへの文字データの書き込み(右寄せ)……113
WRITE#　シーケンシャルファイルへのデータの書き込み……208

キーボード

HELP OFF　HELP キーによる割り込み処理を禁止……73
HELP ON　HELP キーによる割り込み処理を許可……73
HELP STOP　HELP キーによる割り込み処理を停止……73
INKEY$　キーボード入力のチェック……76
INPUT　キーボードからの入力……78
INPUT$　キーボードから文字を読み取る……80
INPUT WAIT　キーボードからの入力(時間制限付き)……81
KEY　ファンクションキーの設定……87
KEY LIST　ファンクションキーの内容を画面に表示……88
KEY OFF　ファンクションキーによる割り込み処理を禁止……89
KEY ON　ファンクションキーによる割り込み処理を許可……89
KEY STOP　ファンクションキーによる割り込み処理を停止……89
KINPUT　日本語入力モードにしてキーボード入力……91
LINE INPUT　キーボードからの一行入力……103
LINE INPUT WAIT　キーボードからの一行入力(時間制限付き)……105
ON HELP GOSUB　HELP キーによる割り込み処理……127
ON KEY GOSUB　ファンクションキーによる割り込み処理……128
ON STOP GOSUB　STOP キーによる割り込み処理……132
STOP OFF　STOP キーによる割り込み処理を禁止……185
STOP ON　STOP キーによる割り込み処理を許可……185
STOP STOP　STOP キーによる割り込み処理を停止……185

スクリーン(テキスト画面)

CLS　画面の消去……24
COLOR[1]　文字の色の設定……25
COLOR@　表示している文字の色を変更……29
CONSOLE　テキスト画面のさまざまなモード設定……32
COPY　画面のハードコピー……34
CSRLIN　カーソルの行位置を知らせる……37
LOCATE　カーソルの位置の設定と表示……109

POS	カーソルの桁位置を知らせる	148
PRINT	データの画面への表示	149
PRINT USING	書式指定による画面への表示	151
WIDTH	表示文字数の設定	203
WRITE	データの画面への表示	207

スクリーン(グラフィック画面)

CIRCLE	円・楕円を描く	20
CLS	画面の消去	24
COLOR[1]	前景色・背景色・表示する色の設定	25
COLOR[2]	パレット番号とカラーコードの対応を変更	27
COPY	画面のハードコピー	34
DRAW	ペンを動かして図形を描く	47
GET@	グラフィックパターンの読み取り	69
LINE	線・四角形を描く	101
MAP	ワールド座標とスクリーン座標の変換	114
PAINT	図形を塗りつぶす	138
POINT	最終参照座標の変更	144
POINT[1](関数)	最終参照座標を知る	145
POINT[2](関数)	指定座標の点の色を知る	146
PRESET	点の消去	156
PSET	点を描く	156
PUT@	グラフィックパターンの表示	158
ROLL	グラフィック画面のスクロール	169
SCREEN	グラフィック画面の表示モードの設定	172
VIEW	ビューポートの設定	198
VIEW(関数)	ビューポートの設定位置を知る	200
WINDOW	ウィンドウの設定	205
WINDOW(関数)	ウィンドウの設定位置を知る	206

プリンタ

COPY	画面のハードコピー	34
LLIST	プログラムリストのプリンタ出力	106
LPOS	プリンタヘッドの現在位置	112
LPRINT	データのプリンタへの出力	149
LPRINT　USING	書式指定によるプリンタへの出力	151
SPC	空白の出力	182
TAB	データの桁揃え	189
WIDTH LPRINT	出力桁数の設定	204

通信回線

CLOSE 通信回線のクローズ …… 23
COM OFF 通信回線による割り込み処理を禁止 …… 30
COM ON 通信回線による割り込み処理を許可 …… 30
COM STOP 通信回線による割り込み処理を停止 …… 30
EOF 受信バッファのデータの有無 …… 56
INPUT# 通信回線からのデータ読み込み …… 79
INPUT$ 通信回線からのデータ読み込み …… 80
LINE INPUT# 一行のデータ読み込み …… 104
LOC 受信バッファのデータ数 …… 108
LOF 受信バッファの空き領域 …… 110
ON COM GOSUB 通信回線による割り込み処理 …… 124
OPEN "COM1 : 通信回線のオープン …… 134
PRINT# 通信回線へのデータ送信 …… 154
PRINT# USING 書式指定によるデータ送信 …… 154
WIDTH 送信データ数の設定 …… 203
WRITE# 通信回線へのデータ送信 …… 208

日付・時刻

DATE$ 日付の設定・読み出し …… 40
ON TIME$ GOSUB 時刻による割り込み処理 …… 133
TIME$ 時刻の設定・読み出し …… 191
TIME$ OFF 時刻による割り込み処理を禁止 …… 192
TIME$ ON 時刻による割り込み処理を許可 …… 192
TIME$ STOP 時刻による割り込み処理を停止 …… 192

機械語プログラム・メモリ

BLOAD 機械語プログラムのロード(読み込み) …… 12
BSAVE 機械語プログラムのセーブ(書き込み) …… 13
CALL 機械語プログラムの実行 …… 14
CLEAR 機械語プログラム領域の設定 …… 22
DEF SEG セグメントアドレスの設定 …… 43
DES USR 機械語プログラムの開始アドレスの設定 …… 44
FRE ユーザーエリアの未使用領域を知る …… 67
MON 機械語モニタの起動 …… 119
PEEK メモリから1バイト読み出る …… 141
POKE メモリに1バイト書き込む …… 147
USR 機械語プログラムの実行 …… 194
VARPTR 変数・ファイル制御ブロックのアドレスを知る …… 196

数値処理・数値関数

ABS 絶対値 …… 5
ATN 逆正接(アークタンジェント) …… 8
CDBL 倍精度実数に変換 …… 15
CINT 整数に変換(小数点以下四捨五入) …… 19
COS 余弦(コサイン) …… 35
CSNG 単精度実数に変換 …… 36
CVD 8バイト文字列を倍精度実数に変換 …… 38
CVI 2バイト文字列を整数に変換 …… 38
CVS 4バイト文字列を単精度実数に変換 …… 38
DEF FN ユーザー定義関数の定義 …… 41
EXP eの指数関数 …… 60
FIX 小数部切り捨て …… 63
INT 小数部切り捨て …… 83
LOG 自然対数 …… 111
RANDOMIZE 乱数シードの変更 …… 160
RND 乱数の発生 …… 168
SGN 正・負符号 …… 179
SIN 正弦(サイン) …… 180
SQR 平方根(ルート) …… 183
TAN 正接(タンジェント) …… 190
VAL 文字列を数値に変換 …… 195

文字列処理・文字関数

ASC 文字の文字コードを知る …… 7
CHR$ 文字コードを文字に変換 …… 18
DEF FN ユーザー定義関数の定義 …… 41
HEX$ 10進数を16進数表記の文字列に変換 …… 74
INSTR 指定した文字列の検索 …… 82
LEFT$ 左から指定バイト数の文字列の取り出し …… 98
LEN 文字列の長さ(バイト数) …… 99
MID$ 文字列の置き換え …… 116
MID$(関数) 文字列の一部を取り出す …… 117
OCT$ 10進数を8進数表記の文字列に変換 …… 123
RIGHT$ 右から指定バイト数の文字列の取り出し …… 167
SPACE$ 空白の文字列の作成 …… 181
STR$ 数値を文字列に変換 …… 186
STRING$ 1文字を指定バイト数分の文字列に変換 …… 187
VAL 文字列を数値に変換 …… 195

日本語文字列

AKCNV$ 1バイト文字を2バイト文字に変換 …… 6
JIS$ 2バイト文字の文字コードを知らせる …… 84
KACNV$ 2バイト文字を1バイト文字に変換 …… 85
KEXT$ 文字列から指定した種類の文字列を抜き出す …… 86
KINPUT 日本語入力モードにしてキーボード入力 …… 91
KINSTR 指定した文字列の検索 …… 92
KLEN 文字列の長さ(文字数) …… 93
KMID$ 日本語文字列の一部を取り出す …… 94
KNJ$ 漢字コードを漢字に変換 …… 95
KPLOAD ユーザー定義の文字パターンの登録 …… 96
KTYPE 指定した位置の文字の種類を知らせる …… 97

エラー処理

ERL エラーの発生した行 …… 58
ERR エラーコード …… 58
ERROR エラーの発生 …… 59
ON ERROR GOTO エラー処理ルーチン …… 125
RESUME エラー処理ルーチンからの復帰 …… 165

ディスク

ATTR$ ドライブ・ファイルの属性を調べる …… 9
DSKF ディスクのトラック数やセクタ数を知る …… 50
DSKI$ ディスクから直接データを読み出す …… 51
DSKO$ ディスクに直接データを書き込む …… 53
SET ドライブ・ファイルに属性を設定する …… 178

ライトペン

ON PEN GOSUB ライトペンによる割り込み処理 …… 129
PEN ライトペンの状態を調べる …… 142
PEN OFF ライトペンによる割り込みを禁止 …… 143
PEN ON ライトペンによる割り込みを許可 …… 143
PEN STOP ライトペンによる割り込みを停止 …… 143

割り込み処理

ON COM GOSUB　通信回線による割り込み処理 …… 124
ON ERROR GOTO　エラー処理ルーチン …… 125
ON HELP GOSUB　HELP キーによる割り込み処理 …… 127
ON KEY GOSUB　ファンクションキーによる割り込み処理 …… 128
ON PEN GOSUB　ライトペンによる割り込み処理 …… 129
ON STOP GOSUB　STOP キーによる割り込み処理 …… 132
ON TIME$ GOSUB　時刻による割り込み処理 …… 133
RESUME　エラー処理ルーチンからの復帰 …… 165
RETURN　割り込み処理ルーチンからの復帰 …… 166

音

BEEP　スピーカを鳴らす …… 11

I/O ポート

INP　入力ポートから1バイト入力 …… 77
OUT　出力ポートへ1バイト出力 …… 137
WAIT　入力ポートの状態待ち …… 201

演算子

演算子については日本語 Disk BASIC「2.6　式と演算子」を参照してください。

＋, －, ＊, ／, ^　算術演算子
¥　整数の除算(商)
MOD　整数の除算(剰余)
＝, <>, ><, <, >, <＝, >＝, ＝<, ＝>　関係演算子
AND　論理積(論理演算子)
EQV　同値(論理演算子)
IMP　包含(論理演算子)
NOT　否定(論理演算子)
OR　論理和(論理演算子)
XOR　排他的論理和(論理演算子)

EPSON PC シリーズ日本語 Disk BASIC V3.0

リファレンスマニュアル
1989年7月　第2版 第1刷発行
セイコーエプソン株式会社
広丘事業所　電子機器事業本部
〒399-07 長野県塩尻市広丘原新田80番地

お問い合わせは、エプソン販売(株)までお願いいたします。